NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

RE-ENGINEERING OF A MISSION CRITICAL SATELLITE COMMUNICATIONS COMPONENT TD-1271B/U

by

Joe T. Hirschfelder
Laurence M. Nixon

March, 1998

Thesis Advisor: Man-Tak Shing

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1998	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE RE-ENGINEERING OF A MISSION CRITICAL SATELLITE COMMUNICATIONS COMPONENT TD1271B/U	5. FUNDING NUMBERS
6. AUTHOR(S) Joe T. Hirschfelder and Laurence M. Nixon	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Legacy software in general, and in the DoD environment in particular, presents an ever-growing maintenance challenge to program managers. The software is cumbersome, written in arcane languages and hosted on aging technology hardware. One of the options that is available to the program manager to alleviate this problem is to re-engineer the existing software product and update it to a newer language software hosted on modern equipment.

We review existing research, select a re-engineering methodology, develop an implementation strategy and then perform a 'case study' examination of this methodology and strategy. For the case study, we take a legacy system, the Navy satellite communications multiplexer, the TD1271B/U Multiplexer, examine its existing documentation, develop a code analysis tool, perform the re-engineering on one of its sub-systems, and analyze the results. We provide observations, recommendations and conclusions on changes, enhancements and pitfalls to the methodology that will be of assistance in future re-engineering efforts of legacy systems.

14. SUBJECT TERMS Reverse Engineering, Re-Engineering, Legacy Systems, TD1271B/U.	15. NUMBER OF PAGES 176
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

Approved for public release; distribution is unlimited.

RE-ENGINEERING OF A MISSION CRITICAL SATELLITE COMMUNICATIONS COMPONENT TD1271B/U

Joe T. Hirschfelder
B.A., San Diego State University, 1979

Laurence M. Nixon
B.A., University of California at San Diego, 1983

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1998

Author: Joe T
Joe T. Hirschfelder

Author: Laurence M
Laurence M. Nixon

Approved by:
Man-Tak Shing, Thesis Advisor

Kathleen M. Nelson
Kathleen Nelson, Second Reader

Dan C Boger
Dan Boger, Chairman, Department of Computer Science

ABSTRACT

Legacy software in general, and in the DoD environment in particular, presents an ever-growing maintenance challenge to program managers. The software is cumbersome, written in arcane languages and hosted on aging technology hardware. One of the options that is available to the program manager to alleviate this problem is to re-engineer the existing software product and update it to a newer language software hosted on modern equipment.

We review existing research, select a re-engineering methodology, develop an implementation strategy and then perform a 'case study' examination of this methodology and strategy. For the case study, we take a legacy system, the Navy satellite communications multiplexer, the TD1271B/U Multiplexer, examine its existing documentation, develop a code analysis tool, perform the re-engineering on one of its sub-systems, and analyze the results. We provide observations, recommendations and conclusions on changes, enhancements and pitfalls to the methodology that will be of assistance in future re-engineering efforts of legacy systems.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

One of the more complex issues associated with a software organization of significant size is the existence of large amounts of legacy software. This software typically was developed years ago, modified extensively by a variety of programmers, and is currently operating on obsolete systems hardware. The programmers that worked on the various modifications have moved on to new projects or left the organization entirely. In most instances, limited or no documentation updates have occurred. These factors all build upon each other until the software is no longer supportable. The hardware is typically old and all of the original processor reserve has been consumed, making the hardware as unsupportable as the legacy software that runs on it. The overall system is so bad that it has but one single saving grace. It works.

The program manager is charged with the responsibility of continuing to provide the services of the program while watching his ability to support the program fading. This leaves the program manager few options.

One option is to live with the system as it exists and continue to 'Band-Aid' any problems as best one can. This tends to be an expedient answer early in the life of the program, but sooner or later, this will no longer be a viable option. At some point in time the program manager will not be able to continue to support this legacy code due to physical constraints or expense.

A second option is to completely replace the existing system. This involves reviewing the current program requirements and documentation, as well as the organization operations and goals, and then to factor in those enhancements that may have forced this change in the first place. Based upon this review, a new set of requirements are generated. From these new program requirements, the program manager will need to select hardware, programming language(s), design architecture, and finally implement the software. This is then followed by a training curve to allow the operator to learn the new system. This approach has several drawbacks. The first is usually a lack of current documentation to start the review from. It is commonplace for organizations to try to limit their maintenance costs at the expense of the quality documentation updates. Another problem are the 'hidden' enhancements; functions that are routinely use by operators, but are not documented as requirements or options even in well maintained documentation. A third is the amount of time needed to implement a full development. And, of course, there is the cost. Often for large systems the costs of a full development run hundreds of thousands of dollars, and it is not unknown for a single program's development to cost millions of dollars.

A third option is to scrap everything except for the working code and perform an in-depth analysis on the code to extract the requirements. This will tell the analyst what the code is actually doing. To this must be added the additional requirements not currently implemented. The state of the legacy code can be a drawback to this method. The success of this method is dependent upon the standards and practices used while maintaining the code. If the code was maintained by a series of programmers, each with their own standards and practices, the analyst may find extracting the information to be quite a challenge and extremely time consuming.

The fourth option is a combination of the last two, extracting the best features of each of the two options to limit the cost and time associated with updating the system. This is implemented by performing a code review and using the existing documentation as a guide to the software's basic architecture. This combination of methods reduces the amount of time required to perform the code analysis, while providing a quality requirements trace.

Each of these options has it's own advantages and drawbacks. Early in the life cycle of a software system it is fairly inexpensive to correct and fix an immediate problem. As more changes are made it becomes more expensive to correct defects or make changes. It is estimated that in a large data processing shop, 70 to 80 percent of the budget is consumed by maintenance of legacy programs [40]. Eventually the legacy system will need to be modernized.

When planning the modernization of a computerized information processing system, there are two basic items to consider: the hardware and the software that comprise the system. In the normal course of events finding a new target platform is the easier process. The speed, power, and memory capacities of computer hardware have been increasing; while hardware size and costs have been steadily falling. Replacing the software itself is another question. As the costs of hardware have fallen, the costs of software development have been rising.

This contrast of costs, cheap hardware and expensive software development, leads the program manager to look at ways of lowering the cost of the software while updating the hardware platform. The first approach would be to take the working software and port it to a new, more powerful platform. This can be done in limited circumstances, but he will normally be restricted to remaining within the existing family of processors (i.e. going from a Motorola 68030 to a Motorola 68040). In the general case, he will run into the problem of incompatibility. The existing legacy software will not run on the processor in the new platform. This places the program manager at the crossroads noted above: which of the four options is best for the project?

In this thesis we will examine the 'option four' method of modernization, that of utilizing both the existing code and documentation to develop a new baseline program hosted on a more modern hardware platform; the process of Re-Engineering.

B. RE-ENGINEERING

There are several definitions for re-engineering in the literature today [41] as described in the following:

- "Software reengineering is any activity that (1) improves one's understanding of software, or (2) prepares or improves the software itself, usually of increased maintainability, reusability, or evolvability".
- "The process of modifying the internal mechanisms of a system or program the data structures of a system or program without changing its functionality."

- "The examination and alteration of a subject system to reconstitute it in a new form and subsequent implementations of that form."
- "Re-engineering changes the underlying technology of a system without affecting the overall function, while reverse engineering is the backward engineering of a system to the specification stage. ... Re-engineering ... is the first step toward reverse engineering"

As can be seen from these quotes, the terminology of the process is not clearly defined. Some believe that Re-engineering is the beginning of the Reverse Engineering process, other that Reverse Engineering is the beginning of the Re-engineering process. For our purpose within this thesis, we will look upon reverse engineering as the systematic review and analysis of existing code to establish an information base from which to re-implement the system i.e., the first step in the re-engineering process.

Even within these somewhat conflicting terms, there are levels of re-engineering. If the only requirement is to "copy" the existing system onto a new platform, it will only be necessary to extract the architecture from the existing code and documentation before you can re-implement the same architecture in a new environment. If you want to update the architecture, then you will need to extract up to the design level before re-coding can start. Depending on the level of re-design you wish to implement you will need to extract to one level higher to allow a baseline for forward engineering at the level you want to implement.

C. OUR THESIS

For our thesis we selected a re-engineering methodology from existing research that we thought had the best application to legacy systems. We then examined this methodology of re-engineering by performing a case study. We took a legacy system, examined the existing documentation, performed the re-engineering on one of it's sub-systems and then analyzed the results. Based on this analysis and other observations we will recommend modifications to the methodology that will assist in future re-engineering efforts. Our legacy code for this procedure is a Navy satellite communications multiplexer: the TD1271B/U Multiplexer.

II. TD1271B/U MULTIPLEXER

A. BACKGROUND

Until the space age, the high frequency (HF) spectrum was the only means of passing messages over long distances by radio. Although other frequency spectrum radios were available, their characteristics limited their use to 'line-of-sight' operations or limited data rates. With the space age came the ability to place an Ultra High Frequency (UHF) radio repeater in a geosynchronous orbit. By placing a transmitting station high above the earth, it's 'line-of-sight' earth coverage was greatly expanded. The characteristics of the UHF Satellite Communications lended itself well to the needs of the Navy. It provided dependable over-the-horizon communications capability at accelerated data rates. The high data rates allow the transfer of data and voice. As the capabilities became more developed and terminals became more prolific, time lags and data delays became more regular. System throughput began to suffer. This left the Navy with a limited number of options. One was to reduce the number of users and the amount of data they were sending. This was an extremely unpopular option to the Fleet. They had grown to depend on the types and levels of data being supplied, and their effectiveness would suffer if any of the data were discontinued. The second option was to acquire more UHF channels. This was not a viable option either. The commercial world was rapidly expanding into the UHF spectrum, and lobbying congress to decrease the number of channels reserved for military use and sell them to commercial interests. The only option left was to find ways to utilize the existing channels more efficiently. To meet the need for more efficient usage of the UHF channels the Navy developed the TD-1271B/U Multiplexer.

The basic philosophy behind the TD1271B/U Multiplexer, or MUX, is to use Timed Division Multiple Access (TDMA) principals to integrated multiple nets, each operating on an independent channel, onto a single channel. By multiplexing several channels onto a single channel more data transmission path options become available.

B. TIME DEVISION MULTIPLE ACCESS (TDMA)

In TDMA, data streams at moderate to slow data rates are input via baseband data ports and buffered. At a point in time, predefined for each data port relative to a time marker, this data is transmitted out at a data rate that is significantly faster that the input data rates. While one baseband port's data is being burst out to the satellite, the other ports data streams are being buffered, awaiting their burst opportunity. Using appropriate bursting speeds, none of the baseband units will suffer any loss of continuity in their data. Each will transmit and receive data just as if on a dedicated channel, without realizing the data was multiplexed. This concept is illustrated in Figure 2-1. The key to this process is being able to transmit at a faster data rate than all of the baseband ports data rates combined. The elapse time between the start of one timing mark and the beginning of the next timing mark is generally referred to as a frame. The period of time allocated for a transmission burst is often referred to as a time slot. Depending on the particular protocol for the system, the TDMA Frame may contain a series of time slots with equal time periods or may have time slots with varying time periods.

C. TD1271B/U MULTIPLEXER

In the TD1271B/U Multiplexer (MUX) operating in the TDMA-1 25 kHz mode, a frame is defined as a period of time equaling 1.38666 seconds.[1] This frame not only has varying time slot periods, but is re-configurable based on users needs. The allocation of time slots is determined by the definition of a three digit (hex) Frame Format code. Each of the digits defines the configuration of one of the three user segments contained in the frame. A typical MUX frame is diagrammed in Figure 2-2.

The MUX waveform accommodates multiple input/output (I/O) bit rates and radio frequency (RF) symbol burst rates, as well as dynamic assignment of users between channels. Operation of the MUX waveform is transparent to the user baseband equipment, except for TDMA frame-time delays.

[1] The TD1271 MUX is also capable of a secondary waveform (TDMA-2 or TDMA-3) that utilizes a 5kHz channel. Discussion of this 5 kHz waveform is beyond the scope of this thesis.

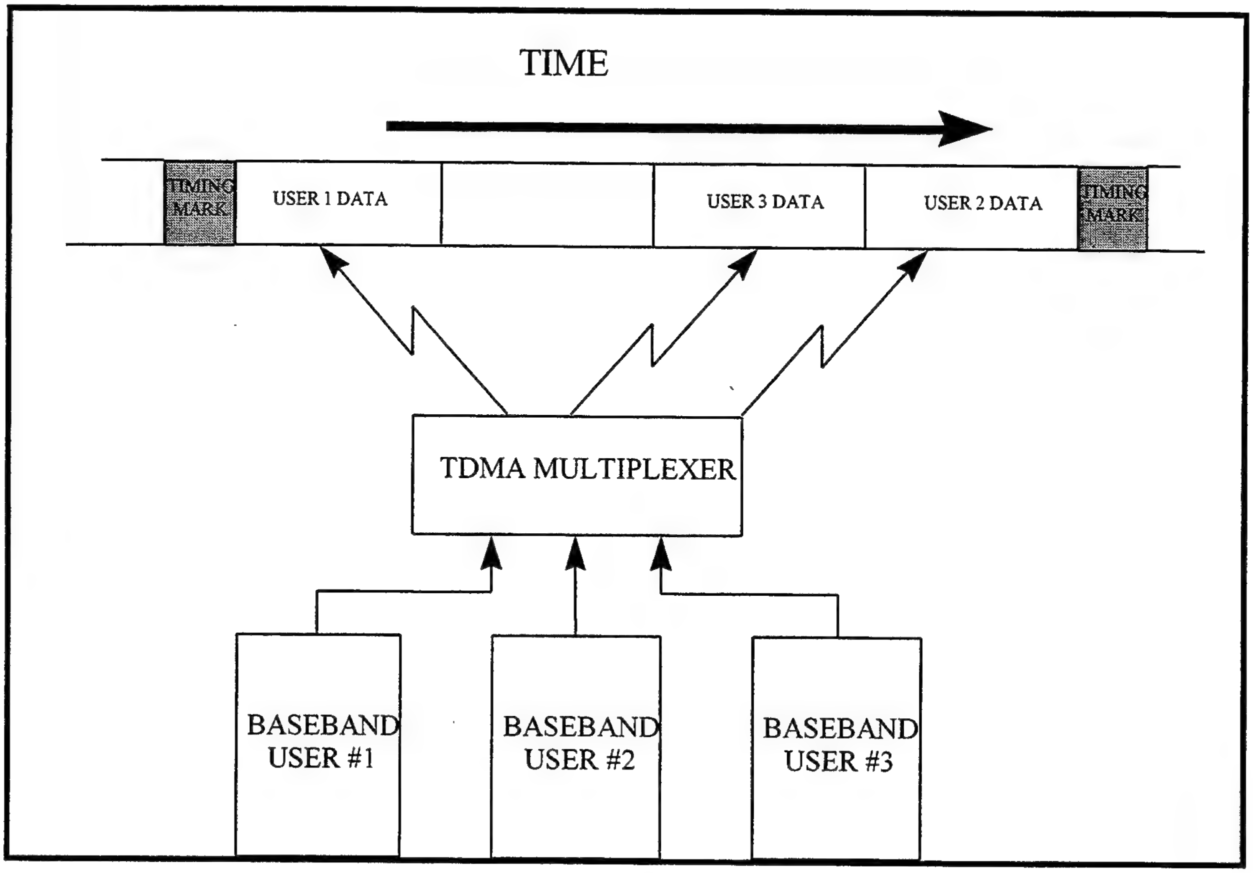


Figure 2-1
Time Division Multiplexing

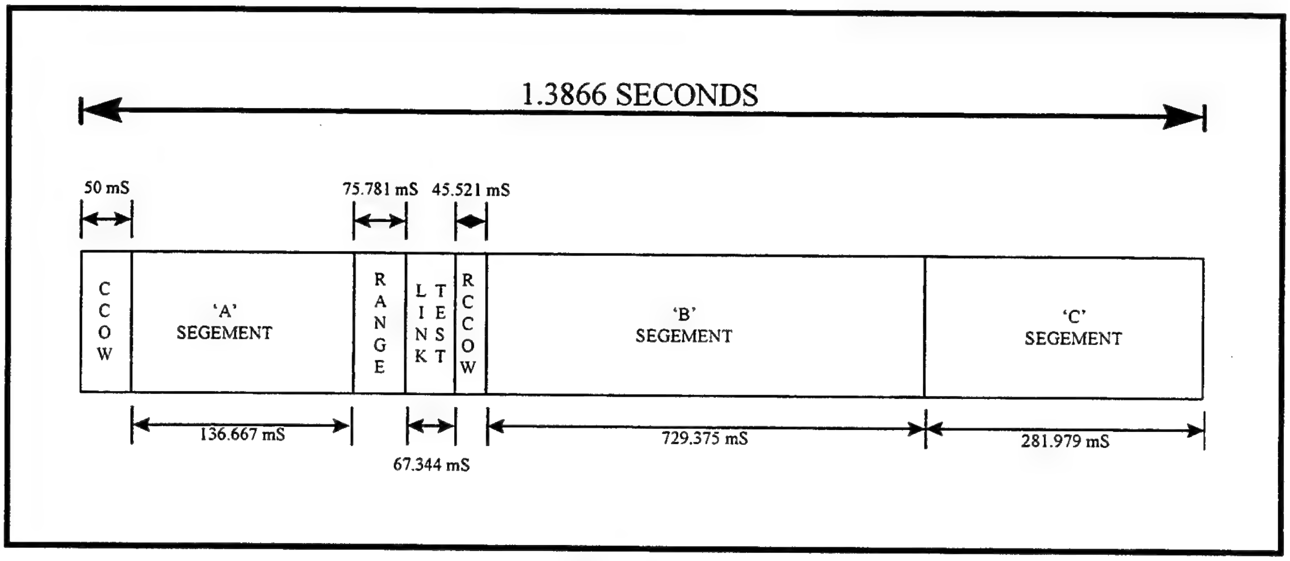


Figure 2-2
Typical TD1271B/U Frame Configuration

The MUX waveform is organized as a repetitive frame structure with 1.386-second frames, as depicted in Figure 2-2 (Frame Type 2 is depicted[2]). Frames are subdivided into slots assigned for: (a) orderwire communications [channel control orderwire (CCOW) and return channel control orderwire (RCCOW)]; (b) system support functions (range time slot and link-test time slot); and (c) user segments A, B, and C. User segments A, B, and C comprise multiple user slots. Each of the 5 slot types in the MUX waveform is identified below:

1. CCOW slot. The CCOW slot is used by the channel controller to broadcast CCOWs. One of the 20 defined CCOWs is transmitted by the Channel Controller in each frame. The CCOW contains the information required to control operation of the waveform and provides a timing mark for the frame.

2. RCCOW slot. The RCCOW slot provides users the opportunity to send data, including requests for access, to the channel controller.

3. Range slot. The range slot provides time for users (including the channel controller) to transmit and receive a data burst for the purpose of calculating the range to the satellite, based on measured round-trip delay time.

4. Link test slot. The link test slot provides time for users (including the channel controller) to transmit and receive a test data stream to determine the current operating conditions, based on the measured bit error ratio (BER). The link test slot is also used to perform dedicated range measurements, when the link test slot is in an even-numbered receive frame. This is known as an even link test (ELT) slot.

[2] In order to support half duplex terminals, the MUX waveform supports a Frame Type 1 that moves the 'C' segment slots between two halves of the 'B' segment. The half duplex network would transmit in the first 'B' segment and receive in the second 'B' segment.

5. User slot. User slots are allocated for transferring data between users. Associated with each user slot is a specific baseband data rate, encoding schema, and burst rate.

Each slot within the MUX waveform consists of three elements: (1) a preamble for receiver carrier, bit, and data synchronization; (2) data; and (3) guard time to avoid user-to-user interference. The receiving equipment achieves carrier and bit synchronization for each slot in the waveform.

The MUX waveform is designed to operate on 25-kHz non-processed channels of current and planned UHF SATCOM satellites.

The MUX accepts input from digital communications systems via it's four baseband data ports. The baseband data ports will accept data at rates of 75, 300, 600, 1200, 2400, 4800, and 16000 bits per second. In order to maintain data integrity, Forward Error Correction (FEC) encoding is used. This FEC coding is convolutional, with interleaving. Encoding rates of ½ or ¾ are used. The MUX will output the data to the radio for transmission at one of three burst rates; 9,600, 19,200 or 32,000 symbols per second. The interface between the radios and the MUX is over 70 MHz intermediate frequency carrier lines. Reception of data is processed in a similar, but reversed, fashion.

One of the most critical functions of the MUX in this process is to maintain consistent timing within the frame. This timing begins with the reception of the timing mark to begin the frame. In the MUX system, all timing is based on the reception of the Channel Control OrderWire (CCOW). The primary CCOW is the Master Frame CCOW and contains the current frame format number, frame sequence number, and cryptographic data needed to define operations on this channel. The Master Frame is transmitted every eight frames. Other CCOWs are available to direct net operations, request data from the various terminals[3] operating on the net and maintain timing.

[3] Although we are only concerned with software and hardware of the TD1271B/U, there are other terminals that are operating on nets with the TD1271B/U. When discussing operations in general, we will use the term terminal to include all of the possible systems operating on a channel.

CCOWs are generated and broadcast by a specifically designated terminal. Each channel will only have a single terminal generating CCOWs at a time (the Primary Channel Controller (PCC)) although other stations could be designated as alternate controllers (ACC).

Once the MUX has successfully received the CCOW, the next critical event is to calculate the terminals distance to the satellite in units of time. The standard time unit in the MUX is the 'time chip'. There are 19,200 time chips in one second. In order for all terminals on the net to receive data in the proper locations within the frame, all transmissions are timed to be correctly placed within the frame at the satellite. In order to insure this 'perfect' frame at the satellite, and by extension at each receiving terminal, the distance to the satellite must be known. This is achieved first at power-up by sending a ranging burst during the designated ranging slot, and updated periodically using the Even Link Test (ELT) slots.

D. TD-1271B/U MULTIPLEXER DESIGN

The basic TD-1271B/U installation comprises a TD-1271B/U multiplexer, a KGV-11 encryption device, a radio transceiver (a Navy WSC-3 pair is typical) and from one to four baseband user devices (ANDVT, ON-143 or teletypewriters would be typical). Figure 2-3 provides a block diagram of a basic TD-1271B/U's configuration.

The software in the MUX is divided into three Computer Software Configuration Items (CSCI); the System Control Processor (SCP), the System Timing Processor(STP) and the Data Buffer Processor (DBP).

Each CSCI has a Memory Program Unit (MPU), an interface board, and at least one 24 k byte Ultra Violet Erasable/Programmable Read Only Memory (UVEPROM) board that contains the executable program. The SCP CSCI utilizes two UVEPROM boards.

The Memory Program Unit (MPU), hosts the Central Processor Unit (CPU), a Motorola 6800, along with the System Random Access Memory.

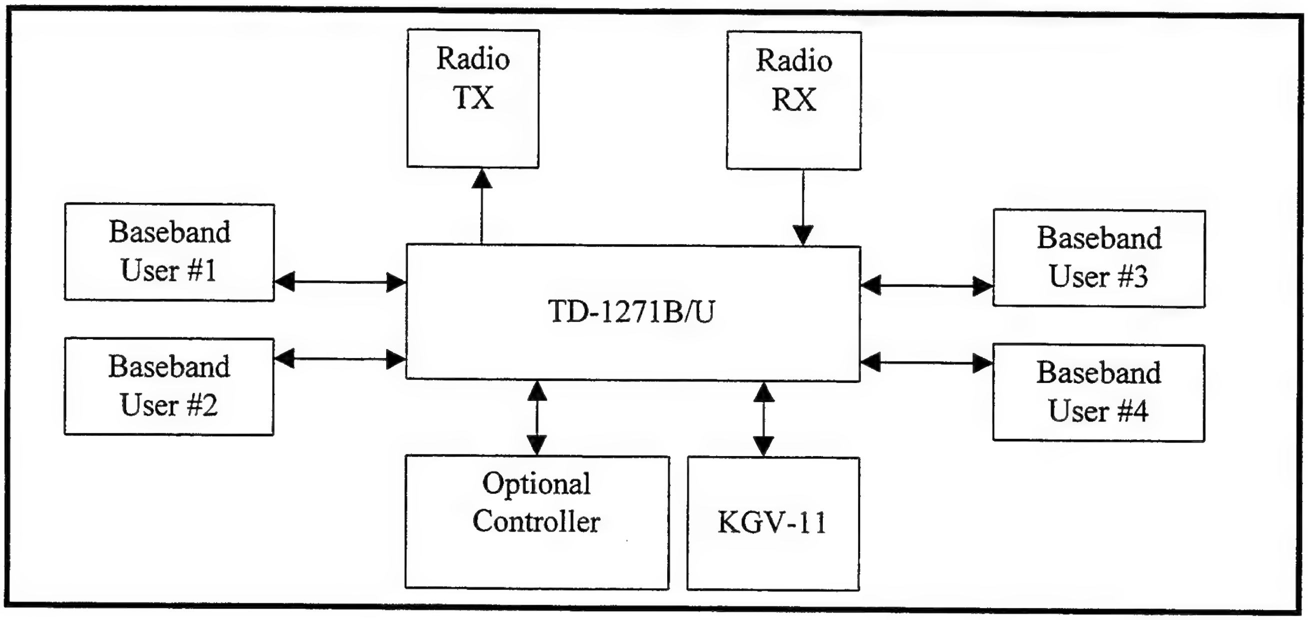


Figure 2-3
A Basic TD-1271B/U Configuration

III. THE RE-ENGINEERING PROCESS

A. PROCESS PHASES

Re-engineering begins with the assembly of all resources of a software system. This includes, but is not limited to the software source code, design documents, software development folders, test reports, software trouble reports, sample inputs, sample outputs, and, if practical, interviews of the implementers and/or maintainers. Typically the only reliable source of information is the software source code and for that reason re-engineering of an existing software system starts at the code level. The software documentation, many times, are not kept current with the software; however, it can still be a useful source of information and can be used as an engineering guide. The process in which existing software is transformed into a new implementation can be described in the Figure 3-1 [30].

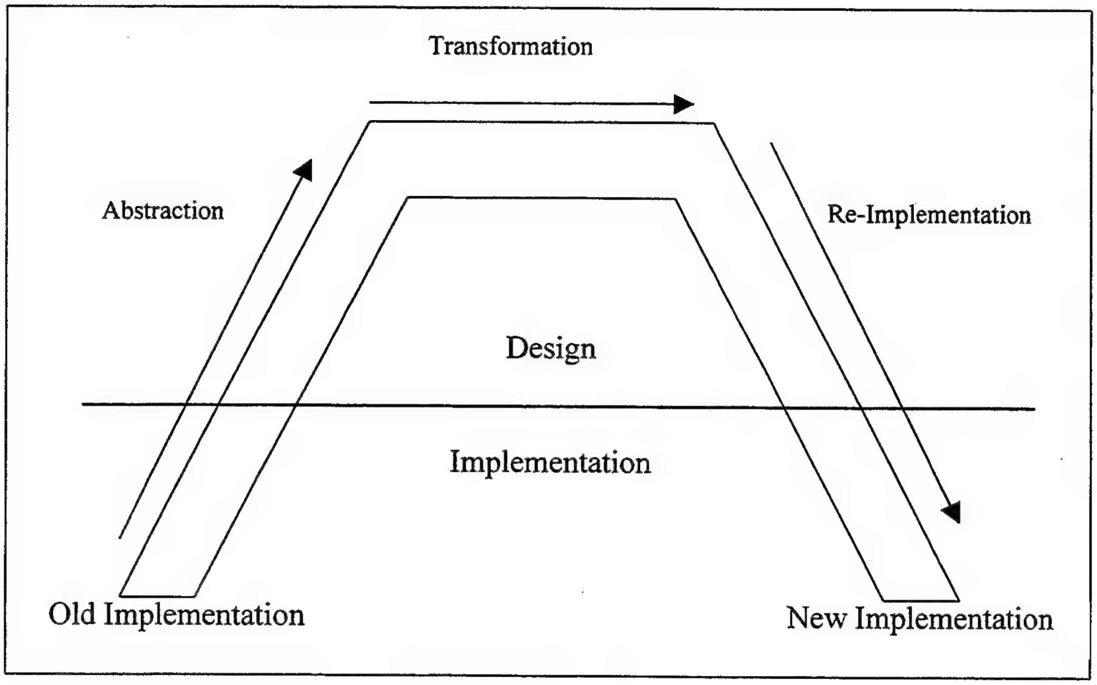


Figure 3-1
The Re-Engineering And Implementation Process

There are three phases to the re-engineering process. The first phase, abstraction, extracts the detailed design information from the documentation and the code. For those familiar with MIL-STD-498, this is analogous to creating a Software Design Document from the existing system. The second phase allows for the modification of the design by adding, deleting, or modifying requirements. The third phase is a re-implementation of the system. The focus of this thesis is the reverse engineering process that recovers the design from the old implementation. This methodology will be validated by executing the re-implementation of the recovered design using a different language and different processor than the old implementation.

1. Phase One: Abstraction

In this phase, the documentation and code can be used to provide the information needed to support the re-creation of the system design. Since the TD-1271 documentation has been shown to be unreliable, the source code will take precedence in this phase with the documentation providing background information. The products developed during abstraction are as follows [30]:

1. Structure Chart. This is a hierarchy of the modules as extracted from the program where each node in the chart corresponds to a procedure or function in the program. Since the goal of abstraction is to remove the language dependencies from the code, it is important to treat library calls as part of the system and include them in the structure chart. The language selected for the re-implementation phase may not provide the same library services. Any language dependent library calls may need to be manually re-implemented. The structure charts for this thesis are based upon the structure charts identified in reference [37]. The structure charts provide a visual representation of the software system that is useful in determining organization and dependencies. This information, once derived, is used to develop the structure charts for the enhanced system.

2. Module Description. This description is a combination of the module's inputs, processing, and outputs, with the algorithms and constraints identified. This description is produced for each module in the structure chart. In addition to a narrative of the module's processing, a language independent form using a specification language such as SPEC [42] or a Program Design Language (PDL) is used to capture the requirements of the module. For this documentation to be useful it is important to verify, through code reading, that the supporting documentation is up-to-date. A case tool that allows this information to be tagged to each element in a structure chart would be very helpful. One of the original goals was to identify a tool to preserve this information as part of this thesis, but due to budgetary considerations no tool was procured.

3. Data Flow Diagram. At a module level the data flow diagram shows the flow of data between the modules. Within the module that the data flow diagram is decomposed, showing at each successive decomposition level the transformations that occur. This information is also used to develop the algorithms that need to be documented when the design is re-implemented. These algorithms can be described using mathematical notation, a Program Design Language, or other notation. These algorithm descriptions are added to the module description in step 2. The process consists of constructing the DFD of level i+1 from level i. Each level explodes the data into transforms, data repositories, and data flows [37]. Data repositories will be represented as boxes, data transforms as circles, and data flows as connecting lines between transforms and data repositories. The transforms are then used in phase 2 to develop the detailed design.

4. Control Flow Diagram. This will identify the control structure of the program between the modules. This information is used to identify the logical relationships between the modules, verses the physical relationship described in step 1. [39] will be used to create this diagram.

5. State Transition Diagram. This will identify the states and modes in which the software can reside and identify the events that cause a change in the state of the software and the processing that is to occur due to the state change, and will identify the modules active for each state. There may be more than one state transition diagram depending upon the concurrency of the system. The notation for this diagram will follow those in [39].

Once the design information has been extracted, the resulting recovered design is reviewed for consistency. Any missing items of information should be investigated and added to the recovered design [30]. While it is the goal of this first phase to extract the design and leave behind the vestiges of the language and platform, the design may still have an organizational structure that was due to the implementation language and platform. Before entering phase two, the extracted design information can be modified to refine the design for portability. For example, the TD1271B/U uses multiprocessors with serial communications between them to perform the processing required. When the system is re-engineered, it is anticipated that this system will be implemented using Ada tasking on a single microprocessor. The detailed design should be abstract enough to make this possible.

2. Phase Two

Phase one used the supporting documents and the code to extract the detailed design information. This phase attempts to link the detailed design information with the requirements through the use of a requirements traceability table. This table binds the modules with the requirements that they partially or fully implement. This table will need to support one to many and many to one. That is, one requirement may be spread out over one or more modules and many requirements may be identified with one module. Depending upon the state of the supporting documentation this can be an easy task or a major effort. Requirements are needed if the system being reengineered is also having the requirements modified as well. Once the requirements have been modified, the modules affected need to be updated. It is not unusual at this phase to add and delete modules from phase one. This phase is identified for completeness only; this thesis will not attempt to construct a requirements traceability table.

The TD1271B/U was developed under DOD-STD-1679 and any re-engineering of the TD1271B/U would use a tailored MIL-STD-498 approach. The purpose of the re-engineering effort is to develop, using the existing resources of the project, those products necessary to pass the exit criteria of the detailed design phase and allow the system to be coded. The primary product that is developed during the detailed design phase is the Software Design Document. When looking at the data item description (DID) the following sections are used to provide the programmer with the information necessary to code the system. From the re-engineering activities performed on the TD1271B/U in the three phases described above, the following module information will be developed:

- Module Name
- Module Description
- Constraints, Limitations, or Unusual Features in the Design
- Special or Unique Algorithm Description
- Called by
- Calls
- Triggering Event (tasks only)
- Implicit Inputs
- Explicit Inputs
- Implicit Outputs
- Explicit Outputs
- Hardware Dependencies
- Sizing and Timing
- Informal Test Procedures

If computer aided design tools were used during phase two, such as the Computer-Aided Prototype System (CAPS), much of the information needed for the SDD is contained within the various reports that the tools produce.

3. Phase Three

At the completion of phase two the design of the current system has been extracted. Before starting the re-implementation of the system for the new target language and hardware, any changes to the design is done in this phase. This could be a re-organization to take advantage of any special capabilities of the compiler or hardware. In addition, this would be the time to modify the requirements of the system and fold those changes into the design. Upon completion of this phase the design should reflect the system as wanted by the customer.

4. Phase Four

The output of phase three is used in this phase. This is the coding of the extracted and possibly modified design. The detail design is re-implemented using the desired methodology (i.e. OOD, structured design, etc.) in the desired language for the desired target platform. This phase is executed in the same manner as any new development.

IV. THE CASE STUDY

A. THE TD1271B/U MULTIPLEXER DATA BUFFERING PROCESSOR

The Data Buffer Processor (DBP) provides the elastic buffering required to pass continuous data between the Systems Control Processor (SCP) and I/O ports while transmitting or receiving data in bursts over the satellite channel. The heart of the DBP consists of 24K bytes of EPROM containing the software program. The program is a loop structure which sequentially scans internal control points and the baseband users to determine the actions is required. If, for example, a user terminal has entered a transmit request at the baseband I/O port, a check is made of the port requirements, the port is configured to receive data and the transmit acknowledge is sent to the baseband device[43]. Figure 4-1 provides a block diagram of the relevant input and output signals to the DBP.[4]

B. DBP INPUT/OUTPUT SIGNALS

The DBP[49] data flow is described in Figure 4-1. Figure 4-1 is a description of data and control signals that are input to or output from the DBP[49].

C. THE RE-ENGINEERING STRATEGY

One of the primary goals of this re-engineering effort was an attempt to use the existing source code to extract the design of the system and to produce well designed, portable code. Ada was used as the target language during the extraction process. Since this software was originally written before the advent of structured design, we didn't attempt to use an Object-Oriented Design approach, but used a structured design approach.

[4] This listing is not complete for the TD1271. Certain hardware control lines are system dependent and would not be applicable to this re-engineering effort.

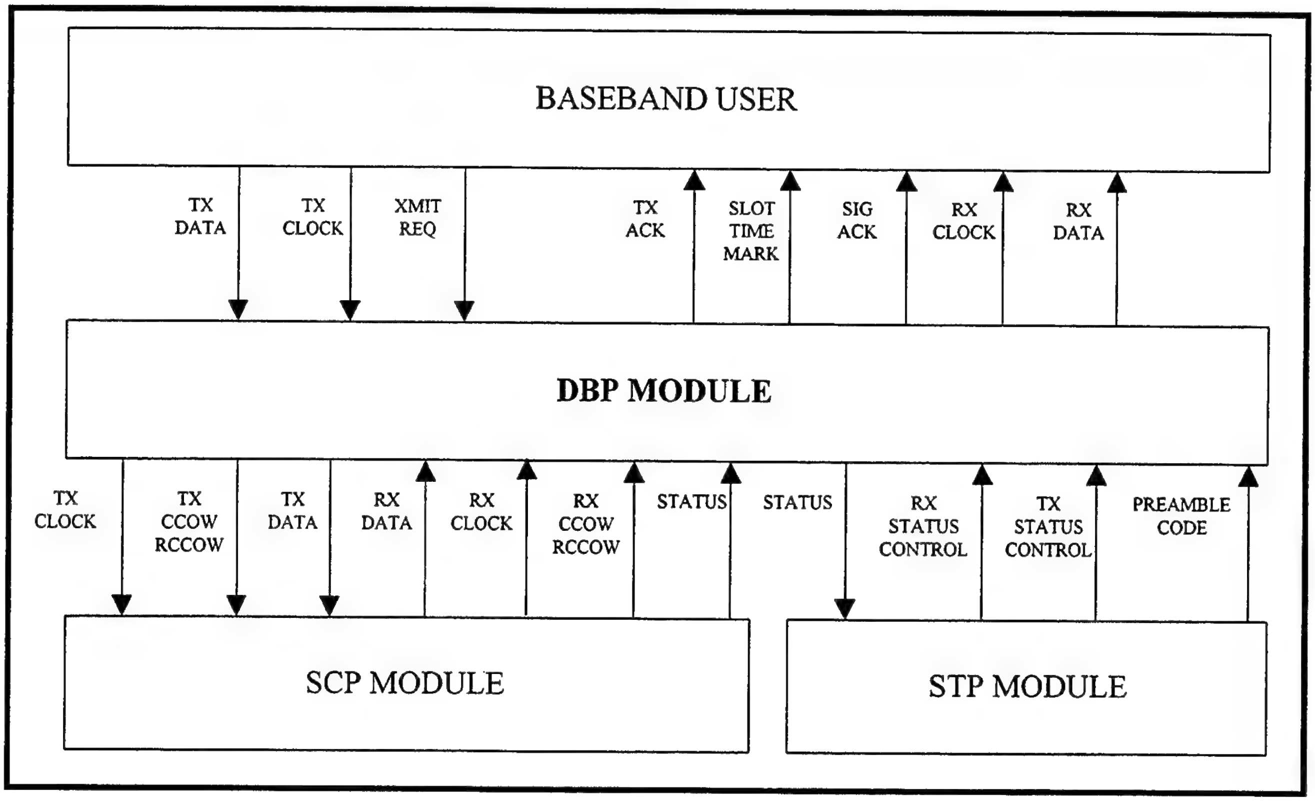


Figure 4-1
Inputs and Outputs To DBP Module

As outlined in Chapter 3, the first phase of the re-engineering effort is to develop Structural Diagram s, Module Descriptions, Data Flow Diagrams, Control Flow Diagrams, and State Transition Diagrams. The code was used to extract the structure, control flow, and state transition diagrams. The documentation was used to develop the module descriptions, provide a high level data-flow diagram and to verify the control flow and state transition diagrams whenever possible.

1. Code review

As noted by T. Bigerstaff [10], a key objective of design recovery is to develop structures that will enable a software engineer to understand the program. Our first step in this process was to identify and partition the code into procedures. This proved to be a task of greater difficulty that we had anticipated. Not only were we dealing with a somewhat cryptic language, we were also faced with a coding style that extensively utilized 'jump' and 'branch' commands to navigate from one set of instructions to another. This programming style was intended to reduce the total size of the code by reusing existing code rather than re-writing it. Programmers coding during this period of time were highly praised if they could reduce the length of the overall program by even a few bytes, regardless of the long term maintenance problems created.

2. Extracting the Structure

The first step was to extract the modules from the code. Modules were defined as a section of code referenced in a Jump to Subroutine (JSR) or Branch to Subroutine (BSR). Even though it was recognized early on that this effort would focus only on part of the TD-1271 system it was important to extract the structure of the entire system to identify the areas in which the rest of the re-engineering effort would focus.

The code for the three major sub-systems were examined and mapped using a Microsoft Excel spreadsheet to record the results (see Appendix B - Subroutine Call Tree). It should be noted that this Subroutine Call Tree is not complete. The TD-1271 is not a procedural based design, that is it makes extensive use of GOTO statements for control between subroutines. The effort to extract the subroutine calls using the JSR/BSR opcodes was sufficient to identify which sub-system to apply the more rigorous code examination to. To produce the Subroutine Call Tree each file comprising the TD-1271 was opened in a word processor and a search was conducted on the words JSR and BSR. At each occurrence the subroutine containing the call and the subroutine that was being called were entered into the Subroutine Call Tree. Duplicate entries, the same subroutine call from the same subroutine, were not noted. The Subroutine Call Tree contains 1183 subroutine calls. After examining the results of this effort the Data Buffer Processor was selected for further investigation.

3. Extracting the Control Flow

In order to extract the control flow from the code it would be necessary to have some sort of cross reference available to consult. There weren't any tools available to do this so one was developed. A parser was written in Ada using an Alsys host compiler for the IBM-PC. This parser uses two passes to build a cross reference for the labels in the source code and a source code listing with line numbers. The first pass identifies each label and the line of code where it is defined. The second pass identifies the line number of code where each reference occurs. References to labels that are not found are reported as errors. The only errors encountered were relative indexes based upon a label (e.g. LABEL1+2). The source code listing is in Appendix D and the Cross Reference Listing is in Appendix E. There will be references to line numbers in the source code from Appendix D later on.

A detailed examination of the DBP code identified the inter-subroutine control flow and a disturbing software design methodology, or to be more accurate - lack of one. As you can see by the control flow diagram, Figure 4-2, the designers used some programming practices that made it very difficult to extract the control flow. The first difficulty encountered was the use of multiple return from subroutine (RTS). Rather than having one entrance and exit from the subroutine there were multiple exits. This was further complicated by the use of GOTO statements where a JSR was used to enter a subroutine and both a RTS and a GOTO statement was used to return (IO1PT to IO1LF in Figure 4-2). There were subroutine calls where the called subroutine executed a GOTO statement to pass control, not back to the caller, but to another routine (IO1PT - IO1LS - IO2PT in Figure 4-2). There were manipulations of the return stack to short circuit the return from a subroutine call (Appendix D, line numbers 1044-1045 and 1099-1100 in IO1PT and repeated for the other three channels). There are examples in which a routine has multiple entrances and exits (Figure 4-3). The way this code was designed made it extremely difficult to convert to Ada, and impossible to meet the goal of having code that is well designed following this conversion. The result was becoming as poorly designed as the original. As stated earlier, this software was designed prior to the introduction of what is now considered modern programming standards. Design decisions were made based more upon memory usage and execution speed than having a system that was easy to maintain.

The subroutine IO1PT was examined for developing the inter-subroutine control flow. This was based upon the conditional statements within the code. This inter-

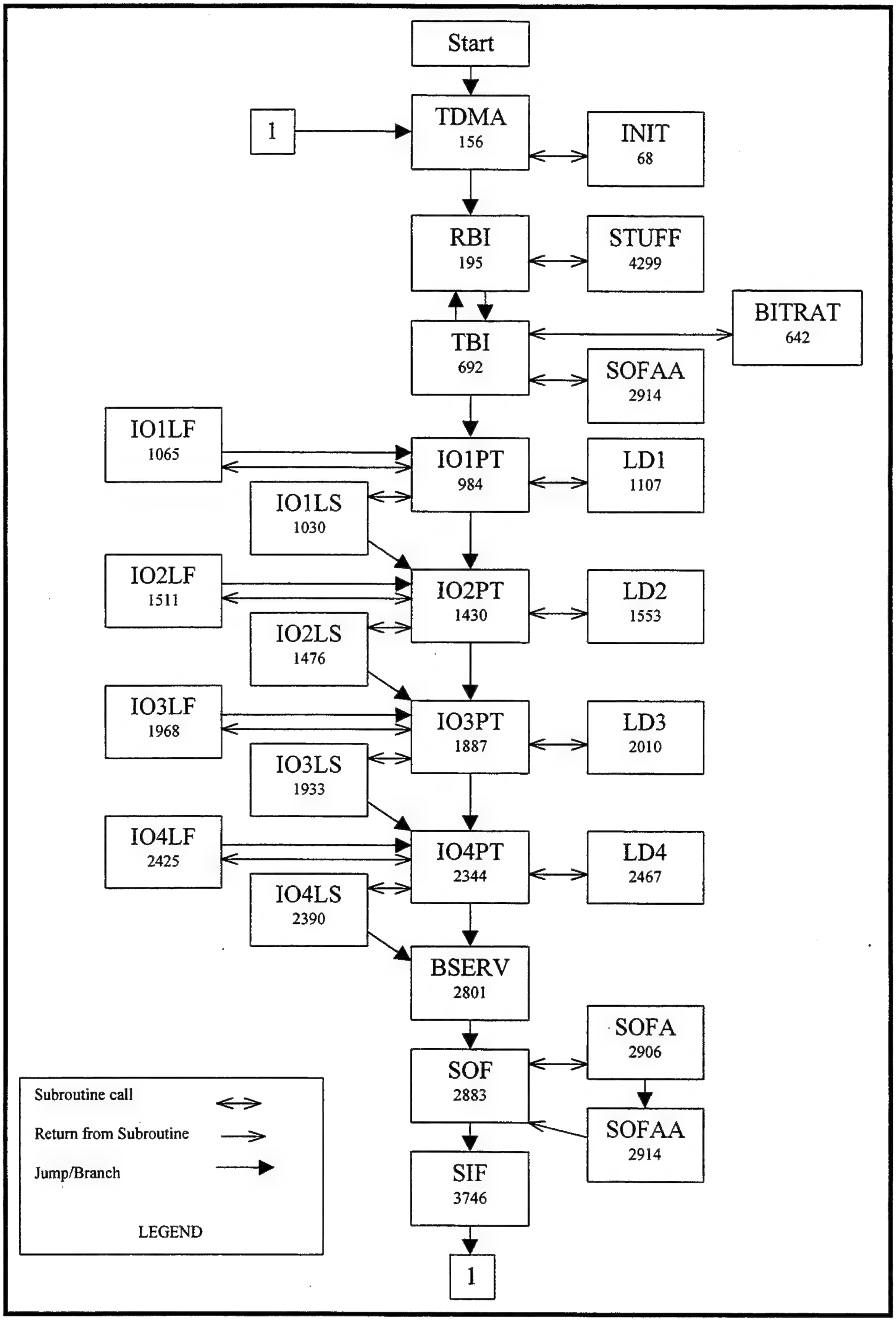
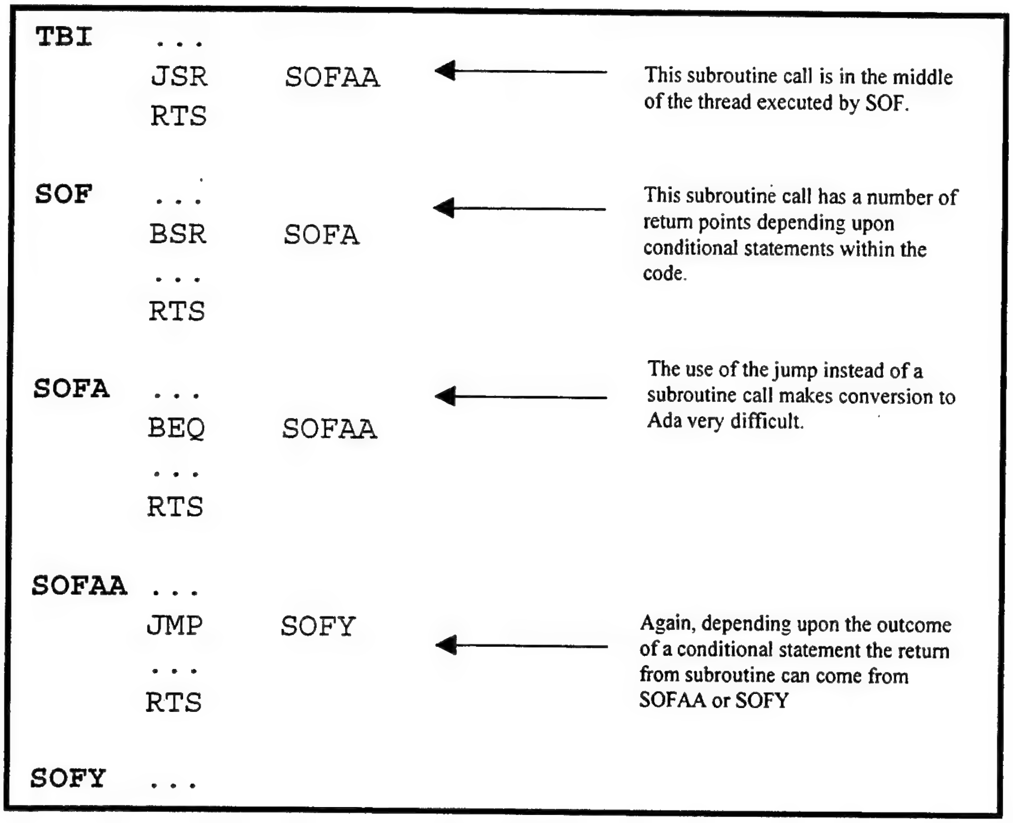


Figure 4-2

DBP Control Flow Diagram



```
TBI    . . .
       JSR     SOFAA
       RTS

SOF    . . .
       BSR     SOFA
       . . .
       RTS

SOFA   . . .
       BEQ     SOFAA
       . . .
       RTS

SOFAA  . . .
       JMP     SOFY
       . . .
       RTS

SOFY   . . .
```

This subroutine call is in the middle of the thread executed by SOF.

This subroutine call has a number of return points depending upon conditional statements within the code.

The use of the jump instead of a subroutine call makes conversion to Ada very difficult.

Again, depending upon the outcome of a conditional statement the return from subroutine can come from SOFAA or SOFY

Figure 4-3

Multiple Entrances and Exits

subroutine control flow was found to be as difficult to convert as the intra-subroutine control flow. The control flow for subroutine IO1A can be visualized with Figure 4-4. As Figure 4-4 depicts, this would be an example of "spaghetti code". In order to overcome this spaghetti code problem it became necessary to reduce the code to 'mini' routines, adding 'jumps' to the end of each code fragment, so each fragment would be a stand alone routine. Each fragment could then be moved at will to try to untangle the subroutine's control flow. It too was far more difficult than it was worth. The result was a subroutine that was poorly designed.

4. Data Flow

The data was analyzed to attempt to produce a data flow diagram. The primary data flow used ringed buffers. The declared buffer started at its own starting address and ended at the starting address of the next declared buffer (e.g. an index pointer was set to

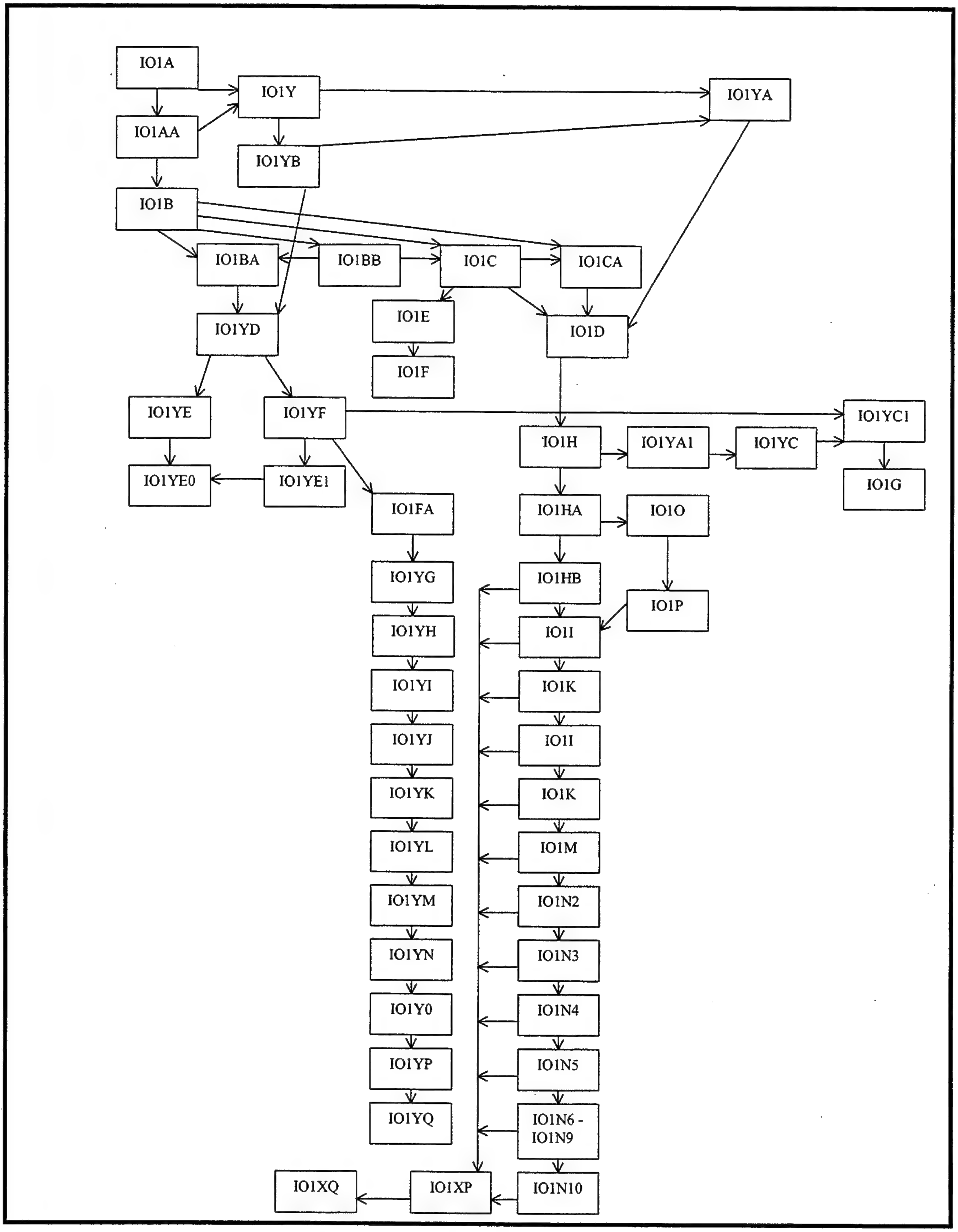


Figure 4-4
IO1A Control Flow

the starting address of a buffer and was incremented until it equaled the starting address of the next buffer). This hard coding is the easiest to overcome, but it is a glaring example of the way the software written. These ringed buffers can be easily defined using a ring buffer generic package in Ada where the data is inserted and extracted using procedure and function calls. Error handling can be included to cover the cases of underflow and overflow. The size of each buffer is then easily established during elaboration based upon the size specified. It was determined that the best source of information to establish a high level data flow diagram was the documentation. This data flow is shown in Figure 4-1. The data flow between the modules of the DBP are primarily indirect by using data stores. It was decided that the effort to describe this data flow was of little use and was not done.

D. USE OF THE TD-1271 AS AN ENGINEERING GUIDE

It was during the effort to establish the control and data flow diagrams that the determination was made that the TD-1271 should be redesigned from the top down. The source code as designed didn't support the re-engineering effort except as an engineering guide during the redesign processes.

The interrupt handlers can be used to establish the tasks responsible for the input and output of data to the DBP subsystem. While the modules are not suitable for re-engineering, the organization of the main modules can be useful in partitioning the new system. You can see by Figure 4-2 that there are four channels that are serviced in a round robin method. The new system can also use this organization. The most valuable asset the current TD-1271 has is by providing a system that works. A comparison between the behavior of the system to various inputs and what is specified in the documentation can give a good indication on the state of the documentation. It provides a strong baseline between stimulus and response that can assist in the formation of the new design.

E. DOCUMENTATION REVIEW

As was noted in the previous section, attempts to extract the requirements directly from the code became a prohibitively tedious and difficult task. This left only the original documentation with which to attempt to baseline the requirements. The primary documentation available for the Data Buffer Processor program includes the Program Performance Specification (PPS) [49], the Program Design Specification (PDS) [50]. Also available are the system level PPS [46] and PDS [47] for the TD1271B/U. The factors and problems noted about documentation in Chapter I are evident in the TD1271B/U's documentation, and bring it's overall accuracy into question. The documents do however provide the basic requirements as they were known as late as 1987. It is these requirements that we shall briefly examine within this section relative to the re-engineering plan detailed in Chapter III.

1. Phase One: Architecture Definition.

a. Structure Chart

The DBP PDS [50, page 3-1] provides a basic structure chart. This is reproduced here as Figure 4-5. This structure chart does not include the modules for initialization and setup.

b. Module Description

Module descriptions for the six 'run time' modules are provided in both the PPS and in the PDS for the DBP. Setup and initialization are not specifically listed. Briefly these six modules are:

1. Receive Burst Initialization Module (RBI). This function processes the receive burst notification from the System Timing Processor. It initializes the DBP's hardware, the I/O ports flags and pointers, and the Service Receive Burst FIFO function.

2. Transmit Burst Initialization Module (TBI). This function processes the receive burst notification from the System Timing Processor. It initializes the DBP's hardware, the I/O ports flags and pointers, and the Service Transmit Burst FIFO function.

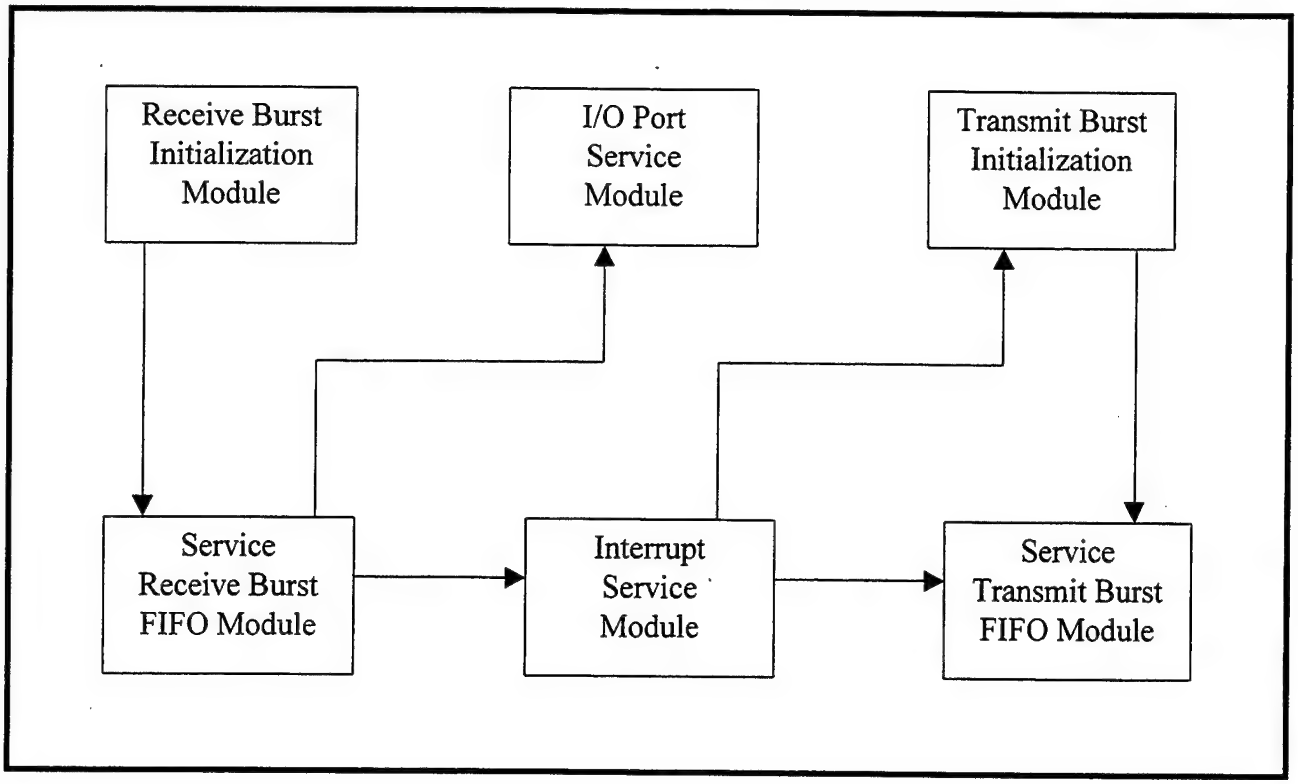

Receive Burst Initialization Module
I/O Port Service Module
Transmit Burst Initialization Module
Service Receive Burst FIFO Module
Interrupt Service Module
Service Transmit Burst FIFO Module

Figure 4-5
DBP High Level Structure Chart

3. <u>I/O Port Service Module (IO1PT-IO4PT)</u>. These four modules services each of the four baseband ports. If the I/O Transmit Request input is true, the DBP will accept I/O port data and store it in the port's data buffer. If the I/O Transmit Request input is false and there is receive burst data for the port, then the I/O port service will output this data from the data buffer to the baseband equipment.

4. <u>Service Receive Burst FIFO Module (SIF)</u>. This module inputs receive burst channel data and stores it in the data buffer.

5. <u>Service Transmit Burst FIFO Module (SOF)</u>. This module outputs the I/O port's transmit data from the data buffer to the burst channel.

6. <u>Interrupt Service Module (IRS)</u>. This module communicates with the System Control Processor. It receives I/O port bit rate and connection status, and it receives/transfers orderwire messages on an interrupt basis [49].

c. Data flow

The Data Flow is somewhat more difficult to extract from the documentation. By definition in the DOD-STD-1679A [51], the full definition of data variables would be documented either in the Program Design Document or in the Database Design Document. To date we have not found any reference to either of these documents. It is our belief that they were tailored out of the documentation package for the original development. This leaves some of the inputs and outputs listed in the DBP PPS and PDS open to question as to their origin and content. Clearly this is an area where it would be necessary to track the variables through out the code to ascertain their design and functions. In addition to the program variables, there are an assortment of Key Lines, Clocks, and Interrupts that need to be processed and input. Figure 4-1 provides a block diagram of the inputs and output from the DBP module.

d. Control flow

From the DBP PDS we were able to extract the basic control flow diagram. This diagram (Figure 4.6) would provide the sequencing of tasks in the new software module. It would also provide the basis for the modules for the Data Flow diagram. In addition to this 'basic' control flow, there is also an Interrupt handling module capable of stopping current processing and shifting control to another module.

2. Phase Two: Requirements Definition.

The second part of the re-engineering effort is to utilize the data gathered from the first phase to develop requirements. The high level requirements can be easily extracted from the module descriptions.

R1: Will receive burst timing from System Timing Processor.

R2: Will initialize hardware and software flags and statuses

R3: Will input data from the Baseband ports and store.

R4: Will receive data from the channel and store.

R5: Will forward input data from storage to the channel.

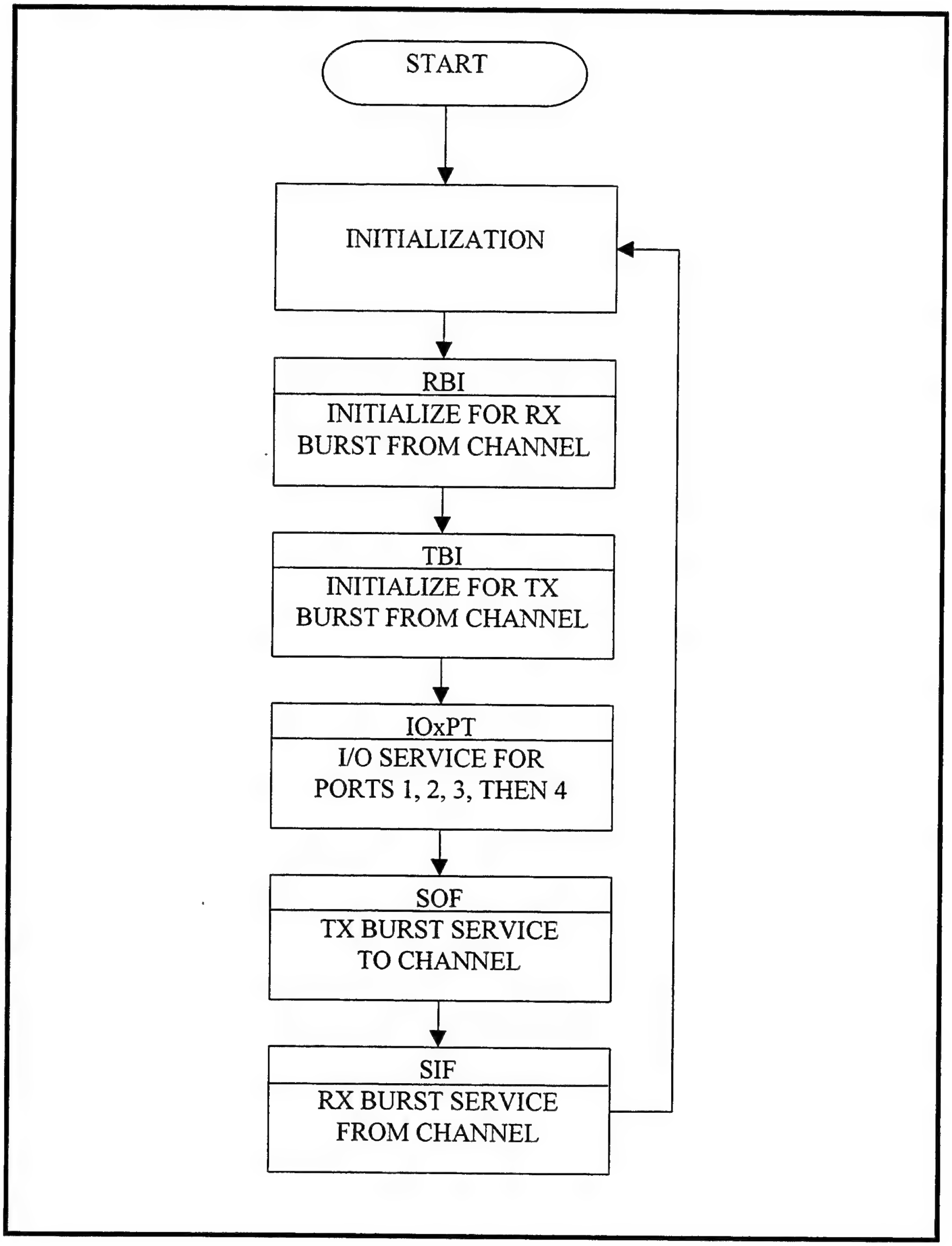


Figure 4-6
Control Flow Diagram

R6: Will output receive data from storage to the Baseband port.

R7: Will receive and transfer orderwires to and from the System Control Processor.

R8: Will receive port status and connection information from the System Control Processor.

From this point, development of more detailed requirements would progress along normal development lines. The advantage that would be gained in this area would be the ability to check the foundation of a requirement in the code. In this way the 'forward' engineering effort would have a existing system to validate the responses against.

F. RECOMMENDATIONS

Effort can be expended to convert the TD-1271 to a high order language and have it run on any number of modern day processors. To do so without reengineering the system from the top down will in effect doom this system to the same fate it has fallen to today: unstructured and not maintainable.

One of the obvious recommendations would be to use a modern programming language. Because of the use of ring buffers between the subsystems, a method that is valid in a new design, Ada would be the best choice. This would allow the use of generic packages to handle the buffers. Ada's tasking and the ability to tie the task's entry point to a hardware interrupt would be another valuable feature that makes Ada the first choice.

The original thesis called for timing studies to be conducted on the re-engineered code. Since the code never reached the state where these studies could be performed there aren't any hard numbers to address. However, the current system runs on three Motorola 6800 microprocessors. The increase in the processors throughput since the Motorola 6800 introduction would give a good indication that the entire system could fit on any number of modern day microprocessors. The hard real time requirement is handling the input from the modem. This data is transmitted at a maximum rate of 16K bits/second. Most real-time embedded applications use Wind River's VxWorks operating system. This operating system allows the application to read a serial port with a variable input size. The programmer specifies the file descriptor for the serial port, the address in which the data is to be read, and the number of bytes to read. In addition, VxWorks provides an optimized task switch for interrupt handlers that has the overhead of a function call. Once the data has been read in the data is output as processed. By proper scaling of the input data from the modem, the processing percentage spent on handling the interrupt can be less than 10%. With these considerations our recommendation would be either the Motorola 68040 or the Motorola Power-PC.

V. CONCLUSIONS

This thesis was originally planned to be a fairly straight forward application and review of the re-engineering process. We very rapidly ran into problems of scope and time. There is a tremendous amount of data written on the subject of re-engineering. The re-engineering data not only covered software, but also included business, hardware, and other re-engineering topics. The first problem we faced was separating out the relevant topics. After 'narrowing' the scope to software re-engineering, we were still faced with a daunting amount of information to review. We found a great deal of information presented on the general theory of re-engineering, but very little concrete information on the application of these theories. We were able to locate a methodology to try [30].

The first phase of this methodology is the extraction of data from the existing code. As noted by Byrne (30), tool support for this portion is a major necessity. With appropriate tools it is possible to statically analyze and document an extensive program in a relatively short period of time. He estimates a 150 to 1 time ratio difference between manual analysis to an automated tool approach. Based on our experience, we would concur with this evaluation. We were unable to locate any tool support to automate the extraction of basic calling trees or other data from the legacy system we were looking at. Although some were found for high order languages, there were very few listings for any Assembly language, and none of these were for the Motorola 6800. This we believe will be prevalent in most legacy systems. Most of the legacy systems will be in some form of Assembly language, but there are not enough in any one language to make it commercially feasible to develop a tool to automate this portion of the re-engineering process.

When faced with poorly designed software it is important to start the re-engineering work at the lowest level in which the design is masked. For example, the preliminary design may be adequate and it was in the execution of the design that the poor design decisions were made. However, it is more likely that the designers, knowing the constraints of the environment in which they have to operate, like available memory, will design to make optimum use of the resources available. There are examples in which the software was written with self-modifying code in order to recover memory space used for initialization as scratch pads for data. However, when re-engineering the system these type of design decisions need to be abandoned and the software needs to be designed to the standards of the times.

With the TD-1271, a tremendous amount of time was used in just trying to understand and decipher the source code. The software, with little or no comments (see Appendix D) and poorly designed control structure, cannot be recommended for re-engineering. Time and effort could be expended to re-engineer the source code: It's not an impossible task, but it is not cost effective. Nor would the result have enhanced the supportability of the software.

The only reasonable approach for the TD1271B/U would be to start at the requirements documentation, establish a testing criteria for each requirement, and perform those tests on the existing system. These tests would demonstrate if there have been modifications to each requirement that would warrant further investigation to determine the current state of that requirement. Once the modifications to the requirements have been made and a suitable target system identified, the task of re-writing the software would be considered a small to medium effort when weighed against current Navy communications systems currently under development.

APPENDIX A

A. BIBLIOGRAPHY

1	“A Conceptual Foundation for Software Re-engineering”, E. Byrne, IEEE 0-8186-2980-0/92
2	“A Framework for Reverse Engineering DoD Legacy Information Systems”, P. Aiken, A. Muntz, R. Richards, IEEE 0-8186-3780-3/93
3	“A Software Re-engineering Process Model”, E. Byrne, D. Gustafson, IEEE 0730-33157/92
4	“A Suzuki Class in Software Reengineering”, Kozaczynski,-Wojtek, IEEE-Software. v. 8 Jan. '91 p. 97-8
5	“An Economic Model to Estimate Software Rewriting and Replacement Times”, Chan,-Taizan; Chung,-Siu-Leung; Ho,-Teck-Hua, IEEE-Transactions-on-Software-Engineering. v. 22 Aug. '96 p. 580-98
6	“Analyzing Empirical Data from a Reverse Engineering Project”, P. Fiore, F. Lanubile, and G. Visaggio, IEEE 0-8186-7111-4/95
7	“Automated Support of Software Maintenance”, K. Bennett, Information And Software Technology, 33(1):74-85, Feb. 1991
8	“Cash Cow in the Tar Pit: Reengineering a Legacy System”, W. Adolph, IEEE Software, May 1996
9	“Concept of the Re-engineering Life-Cycle”, J. Manzella, B. Mutafelija, IEEE 0-8186-2697-6/92
10	“Design Recovery for Maintenance and Reuse”, T. Biggerstaff, Computer, July ‘89
11	“DoD Legacy Systems; Reverse Engineering Data Requirements”, Aiken,-Peter; Muntz,-Alice; Richards,-Russ, Communications-of-the-ACM. v. 37 May '94 p. 26-41
12	“Improving Software Maintenance Through Measurement”, Rombach,-H.-Dieter; Ulery,-Bradford-T, Proceedings-of-the-IEEE. v. 77 Apr. '89 p. 581-95
13	“Investigating Reverse Engineering Technologies for the CAS Program Understanding Project”, Buss,-E; De-Mori,-R; Gentleman,-W.-M, IBM-Systems-Journal. v. 33 no3 '94 p. 477-500
14	“Mandrake: A Tool for Reverse-Engineering IBM Assembly Code”, P. Morris, R. Filman, IEEE Proceedings of WCRE ‘96
15	“Measurement: the key to application development quality”, Walrad,-C; Moss,-E, IBM-Systems-Journal. v. 32 no3 '93 p. 445-60
16	“Planning the Reengineering of Legacy Systems”, Sneed, Harry M., IEEE Software v 12 n 1 Jan 1995
17	“Program and Interface Slicing for Reverse Engineering”, J. Beck D. Eichmann, IEEE 0270-5257/93
18	“Pros and Cons of Reverse Engineering”, http://www.tailormade.com
19	“Recovering Design and Specifications from Source Code”, Prywes,-Noah, IEEE-Software. v. 13 Nov. '96 p. 109-11+
20	“Reengineeing Procedural Into Data Flow Programs”, P. Newcomb, IEEE 0-8186-7111-4/95
21	“Reengineering Definitions”, http://www.stsc.hill.af.mil
22	“Retrieving Information from Data Flow Diagrams”, G. Butler, P. Grogono, R. Shinghal, I. Tjandra, IEEE 0-8186-7111-4/95
23	“Reverse Engineering and Design Recovery: A Taxonomy”, E. Chlkofsky, J. Cross II,

	IEEE Software, Vol. 7, No. 1 Jan. 1990
24	"Reverse Engineering Programs via Dynamic Analysis", H. Ritsch, H. Sneed, Proceedings of WCRE '93
25	"Reverse Engineering through Formal Transformation: Knuths 'Polynomial Addition' Algorithm", Ward,-M.-P, The-Computer-Journal. v. 37 no9 '94 p. 795-813
26	"Reverse Engineering: Algorithms for Program Graph Production", Cimitile,-Aniello; De-Carlini,-Ugo, Software:-Practice-and-Experience. v. 21 May '91 p. 519-37
27	"Software Measurement: A Necessary Scientific Basis", Fenton,-Norman, IEEE-Transactions-on-Software-Engineering. v. 20 Mar. '94 p. 199-206
28	"Software Reengineering: A Quick History", R. Arnold, Communications of the ACM, May 1994
29	"Software Reengineering: Context and Definitions", R. Arnold, Software Reengineering, IEEE Computer Society Press, 1993. Pp. 25
30	"Software Reverse Engineering: A Case Study", Byrne,-Eric-J, Software:-Practice-and-Experience. v. 21 Dec. '91 p. 1349-64
31	"The Evolutionary Growth of Software Reengineering and the Decade Ahead", W. Ulrich, American Programmer, Oct 1990, pp 14-20
32	"Using an Enabling Technology to Reengineer Legacy Systems", Markosian,-Lawrence; Newcomb,-Philip; Brand,-Russell, Communications-of-the-ACM. v. 37 May '94 p. 58-70
33	"Using Data Abstraction to Guide the Restructuring of FORTRAN", R. Ellis, L. Liu, Class handout - source unknown
34	"Using Informal and Formal Techniques for the Reverse Engineeing of C Programs", G. Gannod, B. Cheng, IEEE Proceedings of WCRE '96
35	"Using metrics to evaluate software system maintainability", Coleman,-Don; Ash,-Dan; Lowther,-Bruce, Computer. v. 27 Aug. '94 p. 44-9
36	Untitled reverse engineering article, R. Waters, E. Chikofsky, Communications of the ACM, May 1994
37	"A Reverse Engineering Methodology to Reconstruct Hierarchical Data Flow Diagrams for Software Maintenance", P. Benedusi, A. Cimitile, U. De Carlini, Proceedings of the IEEE Conference on Softwar Maintenance, Oct 1998, pp 180-189
38	A Method for Re-Engineering in Structured Programs", A. Hevner, R. Linger, 22nd Annual Hawaii International Conference on System Sciences, 1989, p. 1025-1034
39	"Object-Oriented Modeling and Design", J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Prentice Hall, 1991.
40	"Re-3' Re-Engineering , Restructuring, Reverse Engineering Part 1", E. Yourdon, American Programmer, Vol 2, No. 4, Apr 1989 p. 3-10, reprinted in R. Arnolds "Software Reengineering" IEEE Computer Society Press, 1992.
41	"A road map guide to Software Reengineering Technology", R. Arnold, Software Reengineering IEEE Computer Society Press, 1992.
42	"Software Engineering with Abstractions", Berzins and Luqi, Addison-Wesley Publishing Company, 1991
43	EE163-GA-OMI-010/E106-TD1271B, Operation And Maintenance Instructions, Organizational Level, Multiplexer TD-1271B/U, Change 2 dtd 07 June 1994,
44	FSCS-211 74, Command And Staff Manual For The UHF DAMA Subsystem, dtd 19 April 1985
45	FSCS-211-24-1 Revision A, UHF Demand Assigned Multiple Access (DAMA) Systems Interface Design Specification For The UHF Satcom Satellite Waveform, dtd 15 January 1985
46	FSCS-212-16 Revision D, UHF Demand Assigned Multiple Access (DAMA) System TD-1271B/U Multiplexer Program Performance Specification, dtd 01 September 1987

47	FSCS-212-17 Revision A, UHF Demand Assigned Multiple Access (DAMA) Systesm TD-1271B/U Multiplexer Program Design Specification, dtd 01 September 1987
48	FSCS-212-26-2 (CH-1), Interface Control Document For The TD-1271B/U Multiplexer (Distributed Control Mode), dtd 30 September 1990
49	FSCS-212-7 Revision A, UHF Demand Assigned Multiple Access (DAMA) Systesm TD-1271B/U Multiplexer Program Performance Specification Data Buffer Processor, dtd 15 May 83
50	FSCS-212-8 Revision A, UHF Demand Assigned Multiple Access (DAMA) Systesm TD-1271B/U Multiplexer Program Design Specification Data Buffer Processor, dtd 01 September 1987
51	DOD-STD-1679A(Navy) Military Standard Software Development, dtd 22 October 1983

APPENDIX B

A. TD-1271B/U SUBROUTINE CALL TREE

Module Name	Located in	Is Called by	Located in
0,X		FPANEL	FP
		SRX	STPV06
		STX0	STPV06
ACK_CC_RCCOW	PRECLA	EXAM_CCOW_OVRHED	PRECLA
ADJUST_FCOUNT_GS	GS	DBP_MSG_INTRPRT	DCEXEC
		DBP_MSG_INTRPRT	PREALT
		EXAM_CCOW_OVRHED	PRECLA
ALARM_ON_GS	GS	FPANEL	FP
		BEEP_ALARM_GS	GS
ALPHA_ONOFF_A_GS	GS	FPANEL	FP
		GLIST_DISPLY_FP	FP
		RECALL_FP	FP
		DC_HOME_FREQ_FP	FP
		AC_HOME_FREQ_FP	FP
		QUEUE_TIME_FP	FP
		CMD_RESPONSE_FP	FP
		FREQ_TO_DSPLY_FP	FP
		KG_TIME_TO_DSPLY	FP
		STATUS_RPT_B_FP	FP
		ZEROIZE_FP	FP
		CONFER_CALL_FP	FP
		INFO_REQUEST_FP	FP
		PAGING_REQ_FP	FP
		KG_MEM_CHANGE_FP	FP
		CONFIG_CODE_FP	FP
		DISP_INFO_REQ_GS	GS
		LAMP_TEST_GS	GS
		LNK_TST_RES_GS	GS
		ERR_GS	GS
		SHOW_PORT_GS	GS
ALPHA_ONOFF_B_GS	GS	FPANEL	FP
		INFO_REPORT_FP	FP
		RECALL_FP	FP
		ZEROIZE_FP	FP
		PORT_RELEASE_FP	FP
		DC_HOME_FREQ_FP	FP
		AC_HOME_FREQ_FP	FP
		SELECT_PORT_FP	FP
		PREC_ENTER_FP	FP
		KG_MEM_FILL_FP	FP
		INFO_REQ_CHK_FP	FP
		EXIT_FP	FP
		LINK_TEST_REQ_FP	FP
		CALL_COMPLETE_FP	FP
		CHAN_CONTROL_FP	FP
		OUT_OF_SERV_FP	FP
		TO_PARITY_CALL_FP	FP
		FORMAT_FREQ_FP	FP
		CONFER_CALL_FP	FP
		KG_TIME_FP	FP

Module Name	Located in	Is Called by	Located in
ALPHA_ONOFF_B_GS	GS	CALL_CANCEL_FP	FP
		INFO_REQUEST_FP	FP
		PAGING_REQ_FP	FP
		KG_MEM_CHANGE_FP	FP
		STG_CONTROL_FP	FP
		CONFIG_CODE_FP	FP
ALPHA_ONOFF_B_GS	GS	LNK_TST_RES_GS	GS
		CALL_CC_GS	GS
		DISP_FCNT_GS	GS
AUDIO_ALARM_01	FP	FPANEL	FP
AUDIO_ALARM_02	FP	INFO_REQ_CHK_FP	FP
AUDIO_ALARM_OFF	FP	FPANEL	FP
		KG_MEM_FILL_FP	FP
BCD_TO_BIN_GS	GS	INFO_REQUEST_FP	FP
		YZ_TO_BIN_FP	FP
		AC_HOME_FREQ_FP	FP
		VWXYZ_TO_BIN_FP	FP
BCD_TO_YZ_FP	FP	FPANEL	FP
		FREQ_TO_DSPLY_FP	FP
		DC_HOME_FREQ_FP	FP
BEEP_ALARM_GS	GS	FPANEL	FP
		INFO_REPORT_FP	FP
		PORT_RELEASE_FP	FP
		DC_HOME_FREQ_FP	FP
		AC_HOME_FREQ_FP	FP
		SELECT_PORT_FP	FP
		PREC_ENTER_FP	FP
		KG_MEM_FILL_FP	FP
		CMD_RESPONSE_FP	FP
		PRE_DCKA_CHK_FP	FP
		CHAN_CONTROL_FP	FP
		LINK_TEST_REQ_FP	FP
		ZEROIZE_FP	FP
		STATUS_RPT_B_FP	FP
		CALL_COMPLETE_FP	FP
		OUT_OF_SERV_FP	FP
		TO_PARITY_CALL_FP	FP
		FORMAT_PARITY_FP	FP
		CONFER_CALL_FP	FP
		KG_TIME_FP	FP
		CALL_CANCEL_FP	FP
		INFO_REQUEST_FP	FP
		PAGING_REQ_FP	FP
		EXTEND_XMIT_FP	FP
		KG_MEM_CHANGE_FP	FP
		STG_CONTROL_FP	FP
		CONFIG_CODE_FP	FP
		PORT_DETECT_GS	GS
		LNK_TST_RES_GS	GS
		ERR_GS	GS
BIN_TO_BCD_CL_GS	GS	FPANEL	FP
		DC_HOME_FREQ_FP	FP
		AC_HOME_FREQ_FP	FP
		QUEUE_TIME_FP	FP
		DISCOTIME_DISPLAY	FP
		FREQ_TO_DSPLY_FP	FP
		KG_TIME_TO_DSPLY	FP

Module Name	Located in	Is Called by	Located in
BIN_TO_BCD_GS	GS	FPANEL	FP
		RECALL_FP	FP
		INFO_REQ_CHK_FP	FP
		FREQ_CODE_CHK_FP	FP
		GLIST_DISPLY_FP	FP
		LNK_TST_RES_GS	GS
BIT_RATE_FIND	DCMOW	CALL_REQUEST	DCMOW
BITE_CLEAR	DCMOW	RRU_INTRPRT	DCMOW
		CALL_REQUEST	DCMOW
		DATA_XFER	DCMOW
		GLIST_DISP	DCMOW
		CALL_COMPLETE	DCMOW
		INFO_REPORT	DCMOW
		RRU_INTRPRT	PREINT
		RRU_STATUS_RPTB	PREINT
		DATA_XFER	PREINT
		GLIST_DISPLAY_05	PREINT
		CALL_COMPLETE	PREINT
		RRU_PRTY_OUT_05	PREINT
		INFO_REPORT_05	PREINT
		CALL_REQ_05	PREINT
		CONFIG_ENTRY_05	PREINT
		RRU_CALLCAN_05	PREINT
		GLIST_ENTDEL_10	PREINT
		PAGING_RCOW	PREINT
		CONF_REQUEST	PREINT
BITRAT	DBPV4	RBIJ	DBPV4
		RBIT	DBPV4
BLINK_SCULITE_GS	GS	EXEC_DCEXEC	DCEXEC
		EXEC_PREALT	PREALT
		CHECKSUM_CHK_GS	GS
BRX	STPV06	CONFIG	STPV06
BSB	STPV06	EXEC	STPV06
		PSA	STPV06
BTX	STPV06	CONFIG	STPV06
CALC_PARITY_GS	GS	XFER_CCOW_TO_DBP	DCEXEC
		IN_CCOW_FROM_DBP	DCEXEC
		ENCRYPT_CCOW	DCEXEC
		DECRYPT_CCOW	DCEXEC
		XFER_RCOW_DBP_GS	GS
		DECRYPT_RCOW_GS	GS
		READ_DBP_RCOW_GS	GS
		ENCRYPT_RCOW_GS	GS
		IN_CCOW_FROM_DBP	PREALT
		ENCRYPT_CCOW	PREALT
		DECRYPT_CCOW	PREALT
		XFER_OW	PREALT
CALL_CC_GS	GS	PORT_DETECT_GS	GS
CALL_COMPLETE_FP	FP	JUMP_COMPLETE_GS	GS
CALL_QUE_UPDATE	PREINT	CALL_IN_QUE_ACK	PREINT
		CALL_IN_QUE	PREINT
CALL_REQ_DCMOW	FP	CALL_REQUEST	DCMOW
		TO_PARITY_CALL_FP	FP
CC_EXEC_DCCNTR	DCCNTR	EXEC_DCEXEC	DCEXEC
CC_EXEC_PRECLA	PRECLA	EXEC_PREALT	PREALT

Module Name	Located in	Is Called by	Located in
CCOW_INTP_DCMOW	DCMOW	EXEC_DCEXEC	DCEXEC
		USEREXEC	DCEXEC
CCOW_INTP_PREINT	PREINT	USEREXEC	PREALT
CHECK_A_RANGE	DCMOW/PREINT	PORT_DISCONNECT	DCMOW
		PORT_DISCONNECT	PREINT
CHECK_B_RANGE	PREINT	PORT_DISCONNECT	DCMOW
		PORT_DISCONNECT	PREINT
CHECK_C_RANGE	DCMOW/PREINT	PORT_DISCONNECT	DCMOW
		PORT_DISCONNECT	PREINT
CHECKSUM_CHK_GS	GS	INIT_DCEXEC	DCEXEC
		INIT	PREALT
CHK_FRAME_COUNT	PRECLA	FRAME_CHK_PRECLA	PRECLA
CHK_GLIST	PREINT	SLOT_DISCONNECT	PREINT
		CALL_WAITING	PREINT
		HOME_CHAN_CHANG	PREINT
		SLOT_CONNECT	PREINT
		FORMAT_CHANGE	PREINT
CLEAR	DBPV4	PWRON	DBPV4
		SETUP	DBPV4
CLEAR_CONT_COUNT	PREINT	BUSY_ACK	PREINT
		CALL_IN_QUE_ACK	PREINT
		USER_OUT_ACK	PREINT
		CALL_ACK	PREINT
CLEAR_DISP_GS	GS	FPANEL	FP
		SHOW_PORT_GS	GS
CLEAR_GNUM	PREINT	RPT_GLIST1	PREINT
		RPT_GLIST2	PREINT
		RPT_GLIST3	PREINT
		RTP_GLIST4	PREINT
CLEAR_GUARD_LIST	DCMOW	FRAME_FMT_UPDATE	DCMOW
CLEAR_LITES_TIME	PREINT	CALL_CANCEL	PREINT
CLEAR_QUE_TIME	PREINT	SLOT_CON_UPDATE	PREINT
		RRU_CALLCAN_15	PREINT
CLEAR_SLOTS_GS	GS	PORT_DISCONNECT	DCMOW
		CHAN_CONTROL_FP	FP
		STATUS_PRECLA	PRECLA
		ZEROIZE	PREINT
		PORT_DISCONNECT	PREINT
CLR_RRU_05	PREALT	RRU_MSG_UPDATE	DCEXEC
CLR_RRU_PREC_DSP	DCEXEC/PREALT	RRU_MSG_UPDATE	DCEXEC
		RRU_MSG_UPDATE	PREALT
CLRTIM	PREINT	BUSY_ACK	PREINT
		USER_OUT_ACK	PREINT
		SLOT_CON_UPDATE	PREINT
		CALL_CANCEL	PREINT
		PARTY_OUT_OF_SYNC	PREINT

Module Name	Located in	Is Called by	Located in
CMD_RESPONSE_FP	FP	ZEROIZE_FP	FP
		INFO_REPORT_FP	FP
		TO_PARITY_CALL_FP	FP
		FORMAT_FREQ_FP	FP
		CONFER_CALL_FP	FP
		DC_HOME_FREQ_FP	FP
		AC_HOME_FREQ_FP	FP
		KG_TIME_FP	FP
		INFO_REQUEST_FP	FP
		GLIST_ENTER_FP	FP
		PAGING_REQ_FP	FP
		KG_MEM_CHANGE_FP	FP
		GLIST_DELETE_FP	FP
		STG_CONTROL_FP	FP
		CONFIG_CODE_FP	FP
		CHAN_CONTROL_FP	FP
COMMAND_FRAME	PREINT	CCOW_INTP_DCMOW	DCMOW
		CCOW_INTP_05	PREINT
COMP_FRAME_GS	GS	FRAME_CHK_DCCNTR	DCCNTR
		FRAME_CHK_PRECLA	PRECLA
		MASTER_FRAME	PREINT
CONFIG_CODE_GS	GS	INFO_REQ_CHK_FP	FP
CONS	STPV06	CONFIG	STPV06
CONT_CHK_PREINT	PREINT	RCOW_PREP_XMIT	PREALT
		CLEAR_CONT_COUNT	PREINT
		READ_CALL_ACK	PREINT
CORRECT_ONE_BCD	GS	BCD_TO_BIN_GS	GS
CORRECT_TWO_BCD	GS	BIN_TO_BCD_GS	GS
CYCLELITES_DCMOW	DCMOW	ZEROIZE	DCMOW
		TS_PREP	DCMOW
		INFO_REQUEST_GS	GS
DBP_MSG_INT_30	DCEXEC/PREALT	EXEC_PREALT	PREALT
DBP_MSG_INT_35	DCEXEC/PREALT	EXEC_DCEXEC	DCEXEC
DBP_MSG_INTRPRT	DCEXEC/PREALT	USEREXEC	DCEXEC
		USERPREAC	DCEXEC
		USERPREAC	PREALT
		USEREXEC	PREALT
DCKPORTOFF_GS	GS	INFO_REQUEST_GS	GS
		PORT_DETECT_GS	GS
DECODE_FP	FP	FPANEL	FP
		KG_MEM_FILL_FP	FP
DECRYPT_CCOW	DCEXEC/PREALT	DBP_MSG_INTRPRT	DCEXEC
		DBP_MSG_INTRPRT	PREALT
DECRYPT_RCOW_GS	GS	DBP_MSG_INTRPRT	DCEXEC
		DBP_MSG_INTRPRT	PREALT
DEL_FROM_GLIST	PREINT	DELET_GUARD_LIST	PREINT
		GLIST_ENTDEL_30	PREINT
DELAY_TOGGLE_GS	GS	STATUS_DCCNTR	DCCNTR
		CC_EXEC_PRECLA	PRECLA
		RCV_CCOW_BLOCK	PRECLA
		STATUS_PRECLA	PRECLA
DID	STPV06	RNGEXC	STPV06
		LTEXEC	STPV06

Module Name	Located in	Is Called by	Located in
DISCO_SLOT_CHK	DCMOW/PREINT	CHECK_A_RANGE	DCMOW
		CHECK_B_RANGE	DCMOW
		CHECK_C_RANGE	DCMOW
		CHECK_A_RANGE	PREINT
		CHECK_B_RANGE	PREINT
		CHECK_C_RANGE	PREINT
DISCOTIME_DISPLY	FP	FPANEL	FP
		RECALL_FP	FP
DISP_FCNT_GS	GS	ZEROIZE_FP	FP
		CHAN_CONTROL_FP	FP
		DC_HOME_FREQ_FP	FP
		AC_HOME_FREQ_FP	FP
		SELECT_PORT_FP	FP
		CALL_COMPLETE_FP	FP
		STATUS_RPT_B_FP	FP
		GLIST_DISPLY_FP	FP
		LINK_TEST_REQ_FP	FP
		INFO_REPORT_FP	FP
		TO_PARITY_CALL_FP	FP
		FORMAT_FREQ_FP	FP
		CONFER_CALL_FP	FP
		KG_TIME_FP	FP
		CALL_CANCEL_FP	FP
		INFO_REQUEST_FP	FP
		GLIST_ENTER_FP	FP
		PAGING_REQ_FP	FP
		KG_MEM_CHANGE_FP	FP
		GLIST_DELETE_FP	FP
		STG_CONTROL_FP	FP
		CONFIG_CODE_FP	FP
		OUT_OF_SERV_FP	FP
		LNK_TST_RES_GS	GS
DISP_INFO_REQ_GS	GS	FPANEL	FP
DISP_VWXYZ_GS	GS	FPANEL	FP
		RECALL_FP	FP
		LNK_TST_RES_GS	GS
DISPLAY_EEEEE_GS	GS	GLIST_DISPLY_FP	FP
		ERR_GS	GS
DISPLY_TO_BCD_GS	GS	FORMAT_FREQ_FP	FP
		AC_HOME_FREQ_FP	FP
		VWXYZ_TO_BIN_FP	FP
		YZ_TO_BIN_FP	FP
		WX_TO_BCD_FP	FP
		YZ_TO_BCD_FP	FP
		CONFIG_CODE_FP	FP
ENCRYPT_CCOW	DCEXEC/PREALT	EXEC_DCEXEC	DCEXEC
		EXEC_PREACT	PREALT
ENCRYPT_RCOW_GS	PREALT	RCOW_PREP_XMIT	DCEXEC
		RCOW_PREP_XMIT	PREALT
EQUIPLED_REDO	FP	FPANEL	FP

Module Name	Located in	Is Called by	Located in
ERR_GS	GS	ZEROIZE_FP	FP
		INFO_REPORT_FP	FP
		TO_PARITY_CALL_FP	FP
		FORMAT_FREQ_FP	FP
		CONFER_CALL_FP	FP
		KG_TIME_FP	FP
		INFO_REQUEST_FP	FP
		GLIST_ENTER_FP	FP
		PAGING_REQ_FP	FP
		EXTEND_XMIT_FP	FP
		GLIST_DELETE_FP	FP
		GLIST_DISPLAY_FP	FP
EVEN_PORT_STORE	PREINT	RRU_STATUS_RPTB	PREINT
EXAM_CCOW_OVRHED	PRECLA	RCV_CCOW_BLOCK	PRECLA
EXIT_FP	FP	TIME_CHK_DCCNTR	DCCNTR
FIND_GLIST	PREINT	ENTER_GUARD_LIST	PREINT
		DELET_GUARD_LIST	PREINT
FIND_GLIST_05	PREINT	GLIST_DISPLAY_05	PREINT
		GLIST_ENTDEL_10	PREINT
		GLIST_ENTDEL_30	PREINT
FIND_PREMPT_ADR	PREINT	PREMPTCOUNT_REDO	PREINT
		SLOT_CON_UPDATE	PREINT
		RRU_DISCO	PREINT
FIND_RRU_BUF_GS	GS	SETUP1	DCMOW
		CALL_REQUEST	DCMOW
		LITES_OFF_DCMOW	DCMOW
		DATA_XFER	DCMOW
		LNK_TST_LAMP_GS	GS
		LNK_TST_RES_GS	GS
		RRU_ACPT_ON_GS	GS
		INFO_REQUEST_GS	GS
		RRU_DISCO	PREINT
		READ_CALL_ACK	PREINT
		CALL_IN_QUE_ACK	PREINT
		USER_OUT_ACK	PREINT
		RRU_OUT_ADR	PREINT
		SLOT_CON_UPDATE	PREINT
		CALL_CANCEL	PREINT
		CALL_WAITING	PREINT
		CALL_IN_QUE	PREINT
		CCOW_DATA_XFER	PREINT
		PARTY_OUT_OF_SVC	PREINT
		LITES_OFF_PREINT	PREINT
		RRU_CALL_CAN_05	PREINT
		DATA_XFER	PREINT
FIND_RRU_QUE	PREINT	DEDICATED_SLOT	PREINT
		BUSY_ACK	PREINT
		CALL_WAITING	PREINT
		FIND_RCOW_QUE	PREINT
		CALL_QUE_UPDATE	PREINT
FIND_RRUQUE_ADR	PREINT	OUT_SVC_UPDATE	PREINT
		CLEAR_QUE_TIME	PREINT

Module Name	Located in	Is Called by	Located in
FIXA_GS	GS	FPANEL	FP
		CONFER_CALL_FP	FP
		QUEUE_TIME_FP	FP
		DISCOTIME_DISPLAY	FP
		KG_MEM_FILL_FP	FP
		FREQ_TO_DSPLY_FP	FP
		PAGING_REQ_FP	FP
		KG_MEM_CHANGE_FP	FP
		LINK_TEST_REQ_FP	FP
		FIXAB_GS	GS
FIXAB_GS	GS	FPANEL	FP
		SHOW_YZ_GS	GS
		SHOW_WX_GS	GS
		FIXA_GS	GS
FIXB_GS	GS	QUEUE_TIME_FP	FP
		DISCOTIME_DISPLAY	FP
		CMD_RESPONSE_FP	FP
		FREQ_TO_DSPLY_FP	FP
		DISPLY_TO_BCD_GS	GS
FMT_GLIST_RCW_05	PREINT	SET_GLIST_PTR	PREINT
FMT_GLIST_RCW_15	PREINT	SET_GLIST_PTR	PREINT
FMT_GLIST_RCW_40	PREINT	SET_GLIST_PTR	PREINT
FMT_GLIST_RCW_55	PREINT	CLEAR_GNUM	PREINT
FORMAT_UPDATE	PREINT	FORMAT_CHANGE	PREINT
FORMAT_GLIST_RCW	PREINT	RPT_GLIST1	PREINT
FP_RCOW	DCEXEC/PREALT	USEREXEC	DCEXEC
		USERPREAC	DCEXEC
		USERPREAC	PREALT
		USEREXEC	PREALT
FPANEL_FP	FP	USEREXEC	DCEXEC
		USERPREAC	DCEXEC
		USEREXEC	PREALT
		USERPREAC	PREALT
FRAME_CHK_DCCNTR	DCCNTR	USEREXEC	DCEXEC
		USERPREAC	DCEXEC
		DBP_MSG_INTRPRT	DCEXEC
FRAME_CHK_PRECLA	PRECLA	USEREXEC	PREALT
		USERPREAC	PREALT
		DBP_MSG_INTRPRT	PREALT
FRAME_FMT_UPDATE	DCMOW	MASTER_FRAME	DCMOW
FREEZE_DISPLAY	FP	FORMAT_FREQ_FP	FP
		DC_HOME_FREQ_FP	FP
		KG_TIME_FP	FP
		KG_MEM_CHANGE_FP	FP
FREQ_CODE_CHK_5	FP	KG_TIME_FP	FP
FREQ_CODE_CHK_FP	FP	FORMAT_FREQ_FP	FP
		DC_HOME_FREQ_FP	FP
FREQ_TO_DSPLY_FP	FP	FORMAT_FREQ_FP	FP
FSLOT	STPV06	SRX	STPV06
GET_RANDOM_NR_GS	GS	RCOW_PREP_XMIT	PREALT
		COMP_RRU_PREC	PREINT

Module Name	Located in	Is Called by	Located in
GET_RCOW_ADR_GS	GS	DATA_XFER	DCMOW
		INFO_REQUEST	DCMOW
		ANY_MSG_IN_QUE	DCMOW
		FRAME_CHK_PRECLA	PRECLA
		RRU_STATUS_RPTB	PREINT
		DATA_XFER	PREINT
		CALL_COMPLETE	PREINT
		RRU_PRTY_OUT_05	PREINT
		INFO_REPORT_05	PREINT
		RRU_CALLCAN_05	PREINT
		RRU_CALL_CAN_15	PREINT
		PAGING_05	PREINT
		FIND_RRU_QUE	PREINT
		ANY_MSG_IN_QUE	PREINT
		CONFIR_LIST_RTP	PREINT
		DEDICATED_SLOT	PREINT
		CLRTIM	PREINT
		CALL_REQ_05	PREINT
		CONF_REQ_05	PREINT
GET_RRU_INADR_GS	GS	RRU_INTRPRT	DCMOW
		CALL_REQUEST	DCMOW
		DATA-XFER	DCMOW
		GLIST_DISP	DCMOW
		CALL_COMPLETE	DCMOW
		INFO_REQUEST	DCMOW
		RRU_INTRPRT	PREINT
		CONF_REQUEST	PREINT
		CALL_REQUEST	PREINT
		RRU_STATUS_RPTB	PREINT
		DATA_XFER	PREINT
		GLIST_DISPLAY	PREINT
		CALL_COMPLETE	PREINT
		RRU_PRTY_OUT_05	PREINT
		INFO_REPORT	PREINT
		RRU_CALL_CANCEL	PREINT
		GLIST_ENTR_DEL	PREINT
		GLIST_ENTDEL_05	PREINT
		PAGING_RCOW	PREINT
		CONFIG_ENTRY	PREINT
GET_TIME_CODE	FP	OUT_OF_SERV_FP	FP
		CONFER_CALL_FP	FP
GLIST_COMPARE	PREINT	GLIST_DISP_05	PREINT
		CHK_GLIST	PREINT
GLIST_COUNTER	PREINT	ENTER_GUARD_LIST	PREINT
		GLIST_ENTDEL_10	PREINT
		GLIST_ENTDEL_30	PREINT
		DELET_GUARD_LIST	PREINT
HI_GUARD_CHK_FP	FP	ZEROIZE_FP	FP
		INFO_REPORT_FP	FP
		TO_PARITY_CALL_FP	FP
		CONFER_CALL_FP	FP
		INFO_REQUEST_FP	FP
		GLIST_ENTER_FP	FP
		PAGING_REQ_FP	FP
IN_CCOW_FROM_DBP	DCEXEC	DBP_MSG_INTRPRT	DCEXEC
		DBP_MSG_INTRPRT	PREALT

Module Name	Located in	Is Called by	Located in
INFO_REPORT_FP	FP	CALL_CC_GS	GS
INFO_REQ_05_GS	GS	HOME_CHAN_05	PREINT
INFO_REQ_CHK_45	FP	SELECT_PORT_FP	FP
INFO_REQ_CHK_50	FP	PORT_RELEASE_FP	FP
INFO_REQ_CHK_FP	FP	FPANEL	FP
INIT	PREALT/DBPV4 (LABEL)	PWRON	DBPV4
		TDMA	DBPV4
INITIAL_ENTRY	PREINT	RRU_STATUS_RPTB	PREINT
		DATA_XFER	PREINT
		CALL_COMPLETE	PREINT
		RRU_PRTY_OUT_05	PREINT
		INFO_RPT_05	PREINT
		RRU_CALL_CAN_15	PREINT
		PAGING_05	PREINT
		STATUS_RPT_A	PREINT
		LINK_RESULT	PREINT
		FORMAT_GLIST_RCW	PREINT
		CALL_REQ_05	PREINT
		CONF_REQ_05	PREINT
IO1LF	DBPV4	IO1A	DBPV4
IO1LS	DBPV4	IO1A	DBPV4
IO2LF	DBPV4	IO2A	DBPV4
IO2LS	DBPV4	IO2A	DBPV4
IO3LF	DBPV4	IO3A	DBPV4
IO3LS	DBPV4	IO3A	DBPV4
IO4LF	DBPV4	IO4A	DBPV4
IO4LS	DBPV4	IO4A	DBPV4
JUMP_COMPLETE_GS	GS	DCKPORTOFF_GS	GS
JUMP_SELECT_1_GS	GS	DCKPORTOFF_GS	GS
		SHOW_PORT_GS	GS
		CALL_CC_GS	GS
JUMP_SELECT_2_GS	GS	DCKPORTOFF_GS	GS
		SHOW_PORT_GS	GS
		CALL_CC_GS	GS
KB_TIME_INIT_GS	GS	ZEROIZE_FP	FP
		OUT_OF_SERV_FP	FP
		STATUS_RPT_B_FP	FP
		GLIST_DISPLY_FP	FP
		LINK_TEST_REQ_FP	FP
		INFO_REQUEST_FP	FP
		TO_PARITY_CALL_FP	FP
		CONFER_CALL_FP	FP
		CALL_CANCEL_FP	FP
		INFO_REQUEST_FP	FP
		GLIST_ENTER_FP	FP
		PAGING_REQ_FP	FP
		CONFIG_CODE_FP	FP
		CALL_COMPLETE_FP	FP
KG_INITIAL_GS	GS	EXEC_DCEXEC	DCEXEC
		EXEC_PREALT	PREALT
		ENCRYPT_CCOW	PREALT

Module Name	Located in	Is Called by	Located in
KG_INITIAL_GS_05	GS	ENCRYPT_CCOW	DCEXEC
		DECRYPT_CCOW	DCEXEC
		DECRYPT_RCOW_GS	GS
		ENCRYPT_RCOW_GS	GS
		ENCRYPT_CCOW	PREALT
		DECRYPT_CCOW	PREALT
KG_LOAD_GS	GS	USERPREAC	DCEXEC
		ENCRYPT_CCOW	DCEXEC
		USERPREAC	PREALT
KG_MEM_CHANGE_GS	GS	EXEC_DCEXEC	DCEXEC
		TS_PREP_DCEXEC	DCEXEC
		EXEC_PREALT	PREALT
		TS_PREP_PREALT	PREALT
		KG_LOAD_GS	GS
KG_MEM_FILL_20	FP	KG_MEM_FILL_FP	FP
KG_MEM_FILL_FP	FP	FPANEL	FP
KG_MEM_TEST_GS	GS	EXAM_CCOW_OVRHED	PRECLA
		KG_MEM_CHANGE_GS	GS
KG_READY_GS	GS	EXAM_CCOW_OVRHED	PRECLA
		KG_RESET_GS	GS
		KG_INITIAL_GS	GS
KG_RESET_GS	GS	EXEC_DCEXEC	DCEXEC
		TS_PREP_DCEXEC	DCEXEC
		ENCRYPT_CCOW	DCEXEC
		DECRYPT_CCOW	DCEXEC
		DECRYPT_RCOW_GS	GS
		ENCRYPT_RCOW_GS	GS
		KG_MEM_CHANGE_GS	GS
		EXEC_PREALT	PREALT
		TS_PREP_PREALT	PREALT
		ENCRYPT_CCOW	PREALT
		DECRYPT_CCOW	PREALT
KG_TIME_75	FP	KG_TIME_FP	FP
KG_TIME_TO_DSPLY	FP	KG_TIME_FP	FP
KG_ZEROIZE_GS	GS	STATUS_DCCNTR	DCCNTR
		STATUS_PRECLA	PRECLA
LD1	DBPV4	IO1A	DBPV4
LD2	DBPV4	IO2A	DBPV4
LD3	DBPV4	IO3A	DBPV4
LD4	DBPV4	IO4A	DBPV4
LITES_OFF_DCMOW	DCMOW	STATUS_DCCNTR	DCCNTR
		ZEROIZE	DCMOW
		FRAME_FMT_UPDATE	DCMOW
		CHAN_CONTROL_FP	FP
LITES_OFF_GS	GS	RECALL_FP	FP
		INFO_REQUEST_FP	FP
		PORT_RELEASE_FP	FP
		INFO_REQ_CHK_FP	FP
		CALL_COMPLETE_FP	FP
		LINK_TEST_REQ_FP	FP
		ZEROIZE_FP	FP
		STATUS_RPT_B_FP	FP
		GLIST_DISPLY_FP	FP
		CHAN_CONTROL_FP	FP
		OUT_OF_SERV_FP	FP

Module Name	Located in	Is Called by	Located in
LITES_OFF_PREINT	PREINT	STATUS_PRECLA	PRECLA
		FORMAT_UPDATE	PREINT
		NUFMT_TO_STP	PREINT
LNK_TST_LAMP_GS	GS	TO_PARITY_CALL_FP	FP
		FORMAT_FREQ_FP	FP
		PORT_RELEASE_FP	FP
		DC_HOME_FREQ_FP	FP
		AC_HOME_FREQ_FP	FP
		QUEUE_TIME_FP	FP
		SELECT_PORT_FP	FP
		CONFER_CALL_FP	FP
		KG_TIME_FP	FP
		CALL_CANCEL_FP	FP
		INFO_REQUEST_FP	FP
		GLIST_ENTER_FP	FP
		PAGING_REQ_FP	FP
		EXTEND_XMIT_FP	FP
		KG_MEM_CHANGE_FP	FP
		GLIST_DELETE_FP	FP
		STG_CONTROL_FP	FP
		CONFIG_CODE_FP	FP
		LAMP_TEST_GS	GS
LO_GUARD_CHK_FP	FP	GLIST_ENTER_FP	FP
LOAD_KG_VARS_GS	GS	ENCYRPT_CCOW	DCEXEC
		DECRYPT_CCOW	DCEXEC
		DECRYPT_RCOW_GS	GS
		ENCRYPT_RCOW_GS	GS
		ENCRYPT_CCOW	PREALT
		DECRYPT_CCOW	PREALT
LTRXEX	STPV06	EXEC	STPV06
MASTER_FRA_05	PREINT	MASTER_FRAME	PREINT
MASTER_FRA_10	PREINT	MASTER_FRAME	PREINT
MASTER_FRAME	DCMOW/PREINT/DCCNTR	CCOW_INTP_DCMOW	DCMOW
		CCOW_INTP_PREINT	PREINT
MOVE_FRAME_GS	GS	MASTER_FRAME	DCCNTR
		FRAME_CHK_DCCNTR	DCCNTR
MOVE_GS	GS	SPECIAL_CCOW	DCCNTR
		RCOW_INTP_DCCNTR	DCCNTR
		INIT_DCEXEC	DCEXEC
		MOVE_TMP_TO_VARS	DCEXEC
		MOVE_VARS_TO_TMP	DCEXEC
		STP_MSG_INTRPRT	DCEXEC
		INIT	PREALT
		MOVE_VARS_TO_TMP	PREALT
		MOVE_TMP_TO_VARS	PREALT
		STP_MSG_INTRPRT	PREALT
		RCOW_INTP_PRECLA	PRECLA
MOVE_TMP_TO_VARS	DCEXEC/PREALT	USERPREAC	PREALT
		USERPREAC	DCEXEC
MOVE_VARS_TO_TMP	DCEXEC/PREALT	USERPREAC	DCEXEC
		USERPREAC	PREALT
NET_NUM_XCHNG	DCEXEC	DECRYPT_CCOW	DCEXEC
NUFMT_TO_STP	PREINT	MASTER_FRAME	PREINT
		FORMAT_CHANGE	PREINT
ODD_PORT_STORE	PREINT	RRU_STATUS_RPTB	PREINT

Module Name	Located in	Is Called by	Located in
OFFSET_BY_3_GS	GS	CC_EXEC_DCCNTR	DCCNTR
		COMMAND_FRAME	DCMOW
OFFSET_BY_6_GS	GS	CC_EXEC_DCCNTR	DCCNTR
		COMMAND_FRAME	DCMOW
OUT_SVC_UPDATE	PREINT	USER_OUT_ACK	PREINT
		PART_OUT_OF_SVC	PREINT
PLACE_CALL_PRTY	PREINT	CONF_REQ_05	PREINT
PLACE_PORT	PREINT	CONFER_LIST_RPT	PREINT
		STATUS_RPT_B	PREINT
POINT_TO_GLIST1	PREINT	SET_GLIST_PTR	PREINT
POINT_TO_GLIST2	PREINT	SET_GLIST_PTR	PREINT
POINT_TO_GLIST3	PREINT	SET_GLIST_PTR	PREINT
POINT_TO_GNUM	PREINT	POINT_TO_GLIST1	PREINT
		POINT_TO_GLIST2	PREINT
		POINT_TO_GLIST3	PREINT
PORT_DETECT_GS	GS	DBP_MSG_INTRPRT	DCEXEC
PORT_DISCO_UPDATE	PREINT	RRU_UPDATE	PREINT
		ZEROIZE	PREINT
PORT_DISCONNECT	DCMOW	FRAME_FMT_UPDATE	DCMOW
		FORMAT_UPDATE	PREINT
		NUFMT_TO_STP	PREINT
PORT_RELEASE_15	FP	FPANEL	FP
PORT_RELEASE_FP	FP	DCKPORTOFF_GS	GS
		SHOW_PORT_GS	GS
		CALL_CC_GS	GS
PORTCCADR_05	PREINT	CONFIG_ENTRY_05	PREINT
PORTCCADR_PREINT	PREINT	CONFIG_CODE_FP	FP
		SLOT_CON_UPDATE	PREINT
		RRU_STATUS_RPTB	PREINT
		CALL_REQ_05	PREINT
		CONF_REQ_05	PREINT
		CONFIG_ENTRY_05	PREINT
PRE_DCKA_CHK_FP	FP	TO_PARTY_CALL_FP	FP
PREC_CONVERT_FP	FP	FPANEL	FP
		TO_PARITY_CALL_FP	FP
PREC_ENTER_25	FP	FORMAT_FREQ_FP	FP
PREC_ENTER_30	FP	GLIST_DISPLY_FP	FP
PREC_ENTER_FP	FP	OUT_OF_SERV_FP	FP
		TO_PARITY_CALL_FP	FP
		CONFER_CALL_FP	FP
PREC_TO_RCOW_GS	GS	DATA_XFER	DCMOW
		DATA_XFER	PREINT
		RRU_PRTY_OUT	PREINT
		CALL_REQ_05	PREINT
		CONF_REQ_05	PREINT
PREEMPT_LITE	PRINT	RRU_DISCO	PREINT
PREEMPT_UPDATE	PRECLA	TIME_CHK_PRECLA	PRECLA
PREMPTCOUNT_05	PREINT	PREMPTCOUNT_REDO	PREINT
PREMPTCOUNT_REDO	PREINT	SLOT_CON_UPDATE	PREINT
		SLOT_DISCONNECT	PREINT
		CALL_COMPLETE	PREINT
PRSN	STPV06	SRX	STPV06
PRT_CNFIG_CHNG	PREINT	RRU_STATUS_RPTB	PREINT
PSA	STPV06	EXEC	STPV06
		CONFIG	STPV06
PTSN	STPV06	STX	STPV06

Module Name	Located in	Is Called by	Located in
PVR	STPV06	SRX	STPV06
		BRX	STPV06
PVRA	STPV06	SRX	STPV06
PVT	STPV06	STX	STPV06
		BTX	STPV06
RAMT	DBPV4/STPV06	COLDS	STPV06
RCCOW_PORTADR_GS	GS	DATA_XFER	DCMOW
		INFO_REPORT	DCMOW
		RRRU_STATUS_RPTB	PREINT
		DATA_XFER	PREINT
		CALL_COMPLETE	PREINT
		RRU_PRTY_OUT_05	PREINT
		INFO_RPT_05	PREINT
		RRU_CALLCAN_15	PREINT
		PAGING_05	PREINT
		STATUS_RPT_A	PREINT
		LINK_RESULT	PREINT
		CALL_REQ_05	PREINT
		CONF_REQ_05	PREINT
RCOW_ASSIGN_FMT	PREINT	CALL_ACK_05	PREINT
		RCOW_ASSIGN_05	PREINT
RCOW_INTP_DCCNTR	DCCNTR	USEREXEC	DCEXEC
RCOW_INTP_PRECLA	PRECLA	USEREXEC	PREALT
RCOW_OUT_PREINT	PRINT	USEREXEC	PREALT
RCOW_PREP_XMIT	DCEXEC/PREALT	USEREXEC	DCEXEC
		USEREXEC	PREALT
RCOW_XMIT_DCMOW	DCMOW	USEREXEC	DCEXEC
RCV_CCOW_BLOCK	PRECLA	CC_EXEC_PRECLA	PRECLA
READ_ACIA	GS	TEST_ACIA	GS
READ_CALL_ACK	PREINT	MASTER_FRAME	PREINT
		COMMAND_FRAME	PREINT
READ_DBP_RCOW_GS	GS	DBP_MSG_INTRPRT	DCEXEC
		DBP_MSG_INTRPRT	PREALT
REDO_DBP_PORT_GS	GS	DISCO_SLOT_CHK	DCMOW
		CALL_REQUEST	DCMOW
		CALL_COMPLETE	DCMOW
		INIT	PREALT
		RRU_TURNOFF	PRECLA
		SLOT_CON_UPDATE	PREINT
		PORT_DISCONNECT	PREINT
		CLEAR_SLOTS_GS	GS
REDO_RRU1_GS	GS	REDO_RRUS_DCMOW	DCMOW
		REDO_RRUS_PREINT	PREINT
		WRITE_ACIA	GS
REDO_RRU2_GS	GS	REDO_RRUS_DCMOW	DCMOW
		REDO_RRUS_PREINT	PREINT
		WRITE_ACIA	GS
REDO_RRU3_GS	GS	REDO_RRUS_DCMOW	DCMOW
		REDO_RRUS_PREINT	PREINT
		WRITE_ACIA	GS
REDO_RRU4_GS	GS	REDO_RRUS_DCMOW	DCMOW
		REDO_RRUS_PREINT	PREINT
		WRITE_ACIA	GS
REDO_RRUS_DCMOW	DCMOW	USEREXEC	DCEXEC
		CCOW_INTP_DCMOW	DCMOW

Module Name	Located in	Is Called by	Located in
REDO_RRUS_PREINT	PREINT	USEREXEC	PREALT
		CCOW_INTP_PREINT	PREINT
REINIT_DELAY	DCMOW	MASTER_FRAME	DCMOW
RESET_ACIAS_GS	GS	INIT_DCEXEC	DCEXEC
		INIT	PREALT
RESET_BLACK_GS	GS	ENCRYPT_CCOW	DCEXEC
		DECRYPT_CCOW	DCEXEC
		DECRYPT_RCOW_GS	GS
		ENCRYPT_RCOW_GS	GS
		KG_INITIAL_GS	GS
		DECRYPT_CCOW	PREALT
		ENCRYPT_CCOW	PREALT
RESET_RRU_PTR_GS	GS	RRU_INTRPRT	DCMOW
		CALL_REQUEST	DCMOW
		DATA_XFER	DCMOW
		GLIST_DISP	DCMOW
		CALL_COMPLETE	DCMOW
		INFO_REPORT	DCMOW
		RRU_INTRPRT	PREINT
		RRU_STATUS_RPTB	PREINT
		DATA_XFER	PREINT
		GLIST_DISPLAY	PREINT
		GLIST_DISPLAY_05	PREINT
		CALL_COMPLETE	PREINT
		RRU_PRTY_OUT	PREINT
		INFO_REPORT	PREINT
		INFO_REPORT_05	PREINT
		CALL_REQUEST	PREINT
		CALL_REQ_05	PREINT
		CONF_REQUEST	PREINT
		RRU_CALL_CANCEL	PREINT
		RRU_CALLCAN_05	PREINT
		GLIST_ENTR_DEL	PREINT
		GLIST_ENTDEL_05	PREINT
		GLIST_ENTDEL_10	PREINT
		PAGING_RCOW	PREINT
		PAGING_05	PREINT
		CONF_REQ_05	PREINT
		CONFIG_ENTRY	PREINT
RGRXEX	STPV06	EXEC	STPV06
ROMTST	DBPV4	PWRON	DBPV4
RPT_B_OUT_PREINT	PREINT	FP_RCOW	PREALT
		STATUS_RPT_B	PREINT
RQST_CODE_GS	GS	INFO_REPORT	DCMOW
		FPANEL	FP
		INFO_REPORT_FP	FP
		INFO_REQUEST_GS	GS
		INFO_RPT_05	PREINT

Module Name	Located in	Is Called by	Located in
RRU_ACPT_ON_GS	GS	DATA_XFER	DCMOW
		CALL_REQUEST	DCMOW
		CALL_COMPLETE	DCMOW
		INFO_REPORT	DCMOW
		RRU_STATUS_RPTB	PREINT
		DATA_XFER	PREINT
		CALL_COMPLETE	PREINT
		RRU_PRTY_OUT_05	PREINT
		INFO_RPT_05	PREINT
		RRU_CALLCAN_15	PREINT
		PAGING_05	PREINT
		CALL_REQ_05	PREINT
		CONF_REQ_05	PREINT
RRU_B2_LOOP_CHK	DCEXEC/PREALT	RCOW_PREP_XMIT	DCEXEC
		RCOW_PREP_XMIT	PREALT
RRU_CLEAR_BIT_GS	GS	STP_MSG_INTRPRT	DCEXEC
		CLR_RRU_PREC_DSP	DCEXEC
		FPANEL	FP
		CLR_RRU_PREC_DSP	PREALT
		STP_MSG_INTRPRT	PREALT
		MASTER_FRAME	PREINT
RRU_DISCO	PREINT	CHK_USER_NR1	PREINT
		CHK_USER_NR2	PREINT
RRU_INTRPRT	DCMOW/PREINT	RRU_MSG_IN_DCMOW	DCMOW
		RRUMSG_IN_PREINT	PREINT
RRU_MSG_IN_DCMOW	DCMOW	USERPREAC	DCEXEC
RRU_MSG_UPDATE	PREALT	USEREXEC	DCEXEC
		USERPREAC	DCEXEC
		USEREXEC	PREALT
		USERPREAC	PREALT
RRU_OUT_ADR	PREINT	BUSY_ACK	PREINT
		CALL_ACK	PREINT
RRU_SET_BIT_GS	GS	STP_MSG_INTRPRT	DCEXEC
		RRU_MSG_UPDATE	DCEXEC
		FPANEL	FP
		STP_MSG_INTRPRT	PREALT
		RRU_MSG_UPDATE	PREALT
		MASTER_FRAME	PREINT
RRU_TURNOFF	PRECLA	TIME_CHK_PRECLA	PRECLA
RRU_XMIT_SET_GS	GS	USEREXEC	DCEXEC
		USERPREAC	DCEXEC
		USEREXEC	PREALT
		USERPREAC	PREALT
RRUMSG_IN_PREINT	PREINT	USEREXEC	PREALT
		USERPREAC	PREALT
RRUOUTAD2_PREINT	PREINT	FRAME_CHK_PRECLA	PRECLA
		BUSY_ACK	PREINT
		CALL-ACK_05	PREINT
		CALL_ACK	PREINT
SAVE_STP11_GS	GS	DISCO_SLOT_CHK	DCMOW
		INIT	PREALT
		RRU_TURNOFF	PRECLA
		SLOT_CON_UPDATE	PREINT
		PORT_DISCONNECT	PREINT

Module Name	Located in	Is Called by	Located in
SAVE_STP12_GS	GS	SWITCH_REDO	DCEXEC
		AC_HOME_FREQ_FP	FP
		EXEC_PREALT	PREALT
		USEREXEC	PREALT
		EXAM_CCOW_OVRHED	PRECLA
		HOME_CHAN_05	PREINT
SAVE_STP13_GS	GS	TIME_CHK_DCCCNTR	DCCNTR
		EXEC_DCEXEC	DCEXEC
		USEREXEC	DCEXEC
		STP_MSG_INTRPRT	DCEXEC
		MASTER_FRAME	DCMOW
		SETUP	PREALT
		USERPREAC	PREALT
		STP_MSG_INTRPRT	PREALT
		EXAM_CCOW_OVRHED	PRECLA
		STATUS_PRECLA	PRECLA
		CC_EXEC_PRECLA	PRECLA
SAVE_STP18_GS	GS	TIME_CHK_DCCCNTR	DCCNTR
		STATUS_DCCNTR	DCCNTR
		USEREXEC	DCEXEC
		USERPREAC	DCEXEC
		STP_MSG_INTRPRT	DCEXEC
		MASTER_FRAME	DCMOW
		RCOW_XMIT_DCMOW	DCMOW
		USEREXEC	PREALT
		STP_MSG_INTRPRT	PREALT
		EXAM_CCOW_OVRHED	PRECLA
		TIME_CHK_PRECLA	PRECLA
		STATUS_PRECLA	PRECLA
		RCOW_OUT_PREINT	PREINT
		MASTER_FRAME	PREINT
		LINK_TEST_ASIGN	PREINT
SAVE_SW3_GS	GS	SETUP_MISC	DCEXEC
		AC_HOME_FREQ_FP	FP
		INIT	PREALT
		SETUP	PREALT
		EXAM_CCOW_OVRHED	PRECLA
		HOME_CHAN_05	PREINT
SCCNV	STPV06	EXEC	STPV06
SEL_PORT_DISCO	PREINT	SLOT_DISCONNECT	PREINT
		RRU_DISCO	PREINT
		SLOT_CON_UPDATE	PREINT
		CALL_COMPLETE	PREINT
SELECT_PORT_FP	GP	JUMP_SELECT_1_GS	GS
		JUMP_SELECT_2_GS	GS
SEND_CCOW_BLOCK	PRECLA	CC_EXEC_PRECLA	PRECLA
SEND_RCOW_TO_SAC	PRECLA	CC_EXEC_PRECLA	PRECLA
SEQUENCE_LITES	DCCNTR	FRAME_CHK_DCCNTR	DCCNTR
SET_DBPSTAT_MSG	DCCNTR/PRECLA	STATUS_DCCNTR	DCCNTR
		STATUS_PRECLA	PRECLA
SET_GLIST_FLAG	PREINT	GLIST_ENTDEL_10	PREINT
		GLIST_ENTDEL_30	PREINT
SET_GLIST_PNTR	PREINT	RPT_GLIST1	PREINT
		RPT_GLIST2	PREINT
		RPT_GLIST3	PREINT
		RPT_GLIST4	PREINT

Module Name	Located in	Is Called by	Located in
SET_MSG_CODE	PREINT	RPT_GLIST1	PREINT
		RPT_GLIST2	PREINT
		RPT_GLIST3	PREINT
		RPT_GLIST4	PREINT
SET_PORT_CC	DCCNTR/PRECLA	STATUS_DCCNTR	DCCNTR
		STATUS_PRECLA	PRECLA
SET_PREMPT_FLAG	PREINT	RRU_DISCO	PREINT
		SLOT_CON_UPDATE	PREINT
SET_SC_FLAG	PREINT	SET_XMIT_FLAG	PREINT
SET_TIME_CODE	PREINT	RRU_DISCO	PREINT
		SLOT_CON_UPDATE	PREINT
SETUP	DBPV4/STPV06	INIT	PREALT
SETUP_MISC	DCEXEC/PREALT	INIT_DCEXEC	DCEXEC
SETUP1	DBPV4/DCMOW	PWRON	DBPV4
		TDMA	DBPV4
SHIFT_EVEN_PORT	PREINT	STATUS_RPT_A	PREINT
SHIFT_ODD_PORT	PREINT	STATUS_RPT_A	PREINT
SHIFTLEFT	GS	LOAD_KG_VARS_GS	GS
SHIFTRIGHT	GS	LOAD_KG_VARS_GS	GS
SHOW_PORT_GS	GS	PORT_DETECT_GS	GS
SHOW_WX_GS	GS	KG_TIME_TO_DSPLY	FP
		DISP_VWXYZ_GS	GS
SHOW_YZ_GS	GS	BCD_TO_YZ_FP	FP
		AC_HOME_FREQ_FP	FP
		KG_TIME_TO_DSPLY	FP
		DISP_VWXYZ_GS	GS
		DISP_INFO_REQ_GS	GS
SHOWOUT_YZ_GS	GS	FPANEL	FP
		CONFIG_CODE_FP	FP
SLOT_CON_UPDATE	PREINT	SLOT_CONNECT	PREINT
SOFA	DBPV4	SOF	DBPV4
SOFAA	DBPV4	RBIT	DBPV4
SRX	STPV06	CONS	STPV06
SST	STPV06	RCW1	STPV06
		RR2	STPV06
STATUS_DCCNTR	DCCNTR	EXEC_DCEXEC	DCEXEC
STATUS_PRECLA	PRECLA	EXEC_PREALT	PREALT
STC	STPV06	CONFIG_CODE_FP	STPV06
STC4	STPV06	STC	STPV06
STG_LITE_UPDATE	FP	STG_CONTROL_FP	FP
STP_MSG_INTRPRT	DCEXEC/PREALT	USEREXEC	DCEXEC
		USERPREAC	DCEXEC
		USEREXEC	PREALT
		USERPREAC	PREALT
STRAP_05	GS	STRAP_READ_GS	GS
STRAP_READ_GS	GS	INIT_DCEXEC	DCEXEC
		SETUP_MISC	DCEXEC
		INIT	PREALT
STUFF	DBPV4	TDMA	DBPV4
STX	STPV06	CONS	STPV06
SUB_KG_TIME_GS	GS	PORT_DETECT_GS	GS
SWAPLITES_DCEXEC	DCEXEC	EXEC_DCEXEC	DCEXEC
		USEREXEC	DCEXEC
		STP_MSG_INTRPRT	DCEXEC
		MASTER_FRAME	DCMOW
		CHAN_CONTROL_FP	FP

Module Name	Located in	Is Called by	Located in
SWITCH_REDO	DCEXEC	EXEC_DCEXEC	DCEXEC
SWITCHIT_GS	GS	EXEC_DCEXEC	DCEXEC
		SWITCH_REDO	DCEXEC
		EXEC_PREALT	PREALT
TEST_ACIA	GS	INTERUPT_GS	GS
TIME_CHK_DCCNTR	DCCNTR	USEREXEC	DCEXEC
		USERPREAC	DCEXEC
TIME_CHK_PRECLA	PRECLA	USEREXEC	PREALT
		USERPREAC	PREALT
TIME_DELAY	DCMOW/PREINT	INFO_REPORT	DCMOW
		RRU_STATUS_RPTB	PREINT
		PAGING_05	PREINT
		INFO_RPT_05	PREINT
		CALL_COMPLETE	PREINT
TIME_FMT	PREINT	RRU_DISCO	PREINT
		SLOT_CON_UPDATE	PREINT
		CALL_QUE_UPDATE	PREINT
		PORT_DISCO_UPDATE	PREINT
TS_PREP_DCEXEC	DCEXEC	FRAME_CHK_DCCNTR	DCCNTR
		USERPREAC	DCEXEC
TS_PREP_PREALT	PREALT	USERPREAC	PREALT
		FRAME_CHK_PRECLA	PRECLA
TST_XMITD_QUE	PREINT	BUSY_ACK	PREINT
		READ_CALL_ACK	PREINT
UNPACK_GS	GS	FPANEL	FP
		INFO_REQ_CHK_FP	FP
		FREQ_TO_DSPLY_FP	FP
		CONFIG_CODE_FP	FP
		SHOW_YZ_GS	GS
		SHOW_WX_GS	GS
USER1_SETUP_GS	GS	ZEROIZE	DCMOW
		INFO_REQUEST_GS	GS
		SLOT_DISCONNECT	PREINT
		PARTY_OUT_OF_SVC	PREINT
		USER1_SETUP_GS	PREINT
		ZEROIZE	PREINT
		CCOW_DATA_XFER	PREINT
		CALL_IN_QUE	PREINT
		CALL_WAITING	PREINT
		DELET_GUARD_LIST	PREINT
		ENTER_GUARD_LIST	PREINT
		HOME_CHAN_CHANG	PREINT
		CALL_CANCEL	PREINT
		FORMAT_CHANGE	PREINT
		SLOT_CONNECT	PREINT
		LINK_TEST_ASIGN	PREINT

Module Name	Located in	Is Called by	Located in
USER2_SETUP	PREINT	ZEROIZE	DCMOW
		SLOT_DISCONNECT	PREINT
		SLOT_CONNECT	PREINT
		FORMAT_CHANGE	PREINT
		CALL_CANCEL	PREINT
		HOME_CHAN_CHANG	PREINT
		ENTER_GUARD_LIST	PREINT
		DELET_GUARD_LIST	PREINT
		CALL_IN_QUE	PREINT
		ZEROIZE	PREINT
		PARTY_OUT_OF_SVC	PREINT
USER3_SETUP	PREINT	CALL_CANCEL	PREINT
		HOME_CHAN_CHANG	PREINT
		ENTER_GUARD_LIST	PREINT
		DELET_GUARD_LIST	PREINT
USEREXEC	DCEXEC/PREALT	EXEC_DCEXEC	DCEXEC
		EXEC_PREALT	PREALT
USERPREAC	DCEXEC/PREALT	EXEC_DCEXEC	DCEXEC
		EXEC_PREALT	PREALT
VWXYZ_TO_BIN_FP	FP	ZEROIZE_FP	FP
		GLIST_DELETE_FP	FP
		PAGING_REQ_FP	FP
		GLIST_ENTER_FP	FP
		INFO_REPORT_FP	FP
		TO_PARITY_CALL_FP	FP
		CONFER_CALL_FP	FP
WRITE_ACIA	GS	TEST_ACIA	GS
WX_TO_BCD_FP	FP	KG_TIME_FP	FP
		INFO_REQUEST_FP	FP
XFER_CCOW_TO_DBP	DCEXEC	EXEC_DCEXEC	DCEXEC
XFER_OW	PREALT	EXEC_PREALT	PREALT
XFER_RCOW_DBP_GS	GS	EXEC_DCEXEC	DCEXEC
		RCOW_PREP_XMIT	DCEXEC
		RCOW_PREP_XMIT	PREALT
XPLUSA_GS	GS	FRAME_CHK_DCCNTR	DCCNTR
		CALL_REQUEST	DCCNTR
		SEQUENCE_LITES	DCCNTR
		RCOW_INTP_DCCNTR	DCCNTR
		RRU_B2_LOOP_CHK	DCEXEC
		SWAPLITES_DCEXEC	DCEXEC
		CALL_REQUEST	DCMOW
		INFO_REQ_CHK_FP	FP
		INIT	PREALT
		RRU_B2_LOOP_CHK	PREALT
		RCOW_INTP_PRECLA	PRECLA
		FORMAT_GLIST_RCW	PREINT
		MASTER_FRAME	PREINT
		RRU_DISCO	PREINT
		CHK_GLIST	PREINT
		XPLUSB_GS	GS
		RRU_CLEAR_BIT_GS	GS
		RRU_SET_BIT_GS	GS

Module Name	Located in	Is Called by	Located in
XPLUSB_GS	GS	FORMAT_FREQ_FP	FP
		DC_HOME_FREQ_FP	FP
		FREQ_TO_DSPLY_FP	FP
		CALL_COMPLETE	DCMOW
		CALL_REQUEST	DCMOW
		BIT_RATE_FIND	DCMOW
YZ_TO_BCD_FP	FP	OUT_OF_SERV_FP	FP
		DC_HOME_FREQ_FP	FP
		KG_TIME_FP	FP
		INFO_REQUEST_FP	FP
YZ_TO_BIN_FP	FP	CONFER_CALL_FP	FP
ZERO_SLOTS_GS	GS	STATUS_DCCNTR	DCCNTR
		ZEROIZE	DCMOW

Module Name	Located in	Is Called by	Located in
ZEROFILL_GS	GS	SPECIAL_CCOW	DCCNTR
		TIME_CHK_DCCNTR	DCCNTR
		STATUS_DCCNTR	DCCNTR
		EXEC_DCEXEC	DCEXEC
		STP_MSG_INTRPRT	DCEXEC
		RCOW_XMIT_DCMOW	DCMOW
		CHAN_CONTROL_FP	FP
		EXEC_PREALT	PREALT
		STP_MSG_INTRPRT	PREALT
		TIME_CHK_PRECLA	PRECLA
		STATUS_PRECLA	PRECLA
		MASTER_FRAME	PREINT
		CONF_REQ_05	PREINT
		REDO_RRU1_GS	GS
		CLEAR_SLOTS_GS	GS

APPENDIX C

A. PARSER.ADA

```
with text_io; use text_io;
procedure parser is

  package int_io is new integer_io(integer);
  use int_io;

  subtype label_range is integer range 1..6;
  subtype opcode_range is integer range 8..13;
  subtype operand_range is integer range 15..20;

  subtype str6  is string(1..6);
  subtype str120 is string(1..120);

  type call_node;
  type call_node_ptr is access call_node;

  type call_node is record
    caller : integer;
    next   : call_node_ptr;
  end record;

  type label_node;
  type label_node_ptr is access label_node;

  type label_node is record
    label     : str6;
    called_by : call_node_ptr;
    line      : integer;
    next      : label_node_ptr;
  end record;

  label_list   : label_node_ptr := null;
  new_label    : label_node_ptr;
  search       : label_node_ptr;
  label        : str6;
  opcode       : str6;
  operand      : str6;
  col_nbr      : natural;
  new_caller   : call_node_ptr;
  input_file   : file_type;
  output_file  : file_type;

  procedure establish_label_list is
    line     : str120;
    line_nbr : integer := 0;
    count    : natural;
    debug    : boolean := false;
    found    : boolean;
  begin
    while not end_of_file(input_file) loop
    get_line(input_file, line, count);
    line_nbr := line_nbr + 1;
    put(output_file,line_nbr,4);
    put_line(output_file," " & line(1..count));
    -- CHECK IF THE LINE HAS A LABEL
    if count > 0 then
      if line(1) >= 'A' and line(1) <= 'Z' then
        if count >= 7 then
          label := line(label_range);
          if debug then put_line("LABEL IS " & label); end if;

          -- CHECK AND SEE IF THE LABEL EXISTS
          search := label_list;
          found  := false;
          while search /= null loop
            if search.label = label then
              found := true;
            end if;
            search := search.next;
          end loop;
          if debug then put_line("SEARCHED FOR LABEL - " &
             boolean'image(found)); end if;
          if not found then
            -- ENTER THE LABEL INTO THE END OF THE LINKED LIST
            new_label       := new label_node;
            new_label.label := label;
            new_label.next  := null;
            new_label.line  := line_nbr;
            search := label_list;
            if search = null then
              -- FIRST ITEM IN THE LIST
              label_list := new_label;
            else
              while search.next /= null loop
                search := search.next;
              end loop;
              search.next := new_label;
            end if;
          end if;
        end if; -- count > 7
      end if; -- 'A' or 'Z'
    end if; -- count > 0
  end loop;
end establish_label_list;

procedure parse_control is
  line       : str120;
  count      : natural;
  line_count : integer := 0;
  debug      : boolean := false;
  found      : boolean;
  caller     : call_node_ptr;

  function control_opcode(opcode : str6) return boolean is
    debug : boolean := false;
  begin
    if debug then put(opcode); end if;
```

```
      if opcode = "BCC   " or
         opcode = "BCS   " or
         opcode = "BEQ   " or
         opcode = "BGE   " or
         opcode = "BGT   " or
         opcode = "BHI   " or
         opcode = "BLE   " or
         opcode = "BLS   " or
         opcode = "BLT   " or
         opcode = "BMI   " or
         opcode = "BNE   " or
         opcode = "BPL   " or
         opcode = "BRA   " or
         opcode = "BSR   " or
         opcode = "BVC   " or
         opcode = "BVS   " or
         opcode = "JMP   " or
         opcode = "JSR   " then
        if debug then put_line(" - TRUE"); end if;
        if debug then skip_line; end if;
        return true;
      else
        if debug then put_line(" - FALSE"); end if;
        if debug then skip_line; end if;
        return false;
      end if;
    end control_opcode;

  begin
    while not end_of_file(input_file) loop
      get_line(input_file, line, count);
      for i in count + 1 .. line'last loop
        line(i) := ' ';
      end loop;

      line_count := line_count + 1;
      if debug then put_line(line(1..count)); end if;
      -- CHECK IF THE LINE ISN'T A COMMENT
      if line(1) /= '*' then
        -- NOT A COMMENT

        opcode  := line(opcode_range);
        operand  := line(operand_range);

        if control_opcode(opcode) then
          if debug then put_line("FOUND BRANCH OPCODE - LOOKING FOR " &
operand); end if;
          -- SEARCH THE LABEL LIST FOR THE OPERAND
          search := label_list;
          found := false;
          while not found loop
            --if debug then put_line("COMPARING " & operand & " TO " &
search.label); end if;
            if operand = search.label then
              -- FOUND MATCH, ADD CURRENT LINE NUMBER TO LIST
              if debug then put_line("FOUND MATCHING LABEL - ADDING
LINE NUMBER"); end if;
```

```
              -- ADD TO END
              new_caller       := new call_node;
              new_caller.next  := null;
              new_caller.caller:= line_count;
              if search.called_by = null then
                search.called_by := new_caller;
              else
                caller := search.called_by;
                while caller.next /= null loop
                  caller := caller.next;
                end loop;
                caller.next := new_caller;
              end if;
              found := true;
            else
              search := search.next;
              if search = null then
                put_line("COULD NOT FIND LABEL " & operand & " AT LINE
" & integer'image(line_count));
                found := true;
              end if;
            end if;
          end loop;
        end if;
      end if;
      if debug then skip_line; end if;
    end loop;
  end parse_control;

  procedure report_results is
  begin
    while label_list /= null loop
      put(label_list.label & " (");
      if label_list.line < 10 then
        put(label_list.line, 1);
      elsif label_list.line < 100 then
        put(label_list.line, 2);
      elsif label_list.line < 1000 then
        put(label_list.line, 3);
      else
        put(label_list.line, 4);
      end if;
      put(") ");
      while label_list.called_by /= null loop
        put(" ");
        put(label_list.called_by.caller, 4);
        label_list.called_by := label_list.called_by.next;
      end loop;
      label_list := label_list.next;
      new_line;
    end loop;
  end report_results;

begin
  open(input_file, in_file, "s_dbp.src");
  create(output_file, out_file, "s_dbp.lst");
  establish_label_list;
  close(input_file);
  close(output_file);
  open(input_file, in_file, "s_dbp.src");
  parse_control;
  close(input_file);
  report_results;
end parser;
```

APPENDIX D

A. DATA BUFFER PROCESSOR SOURCE CODE LISTING

```
***********************************************************
*
*      BEGIN POWER ON INIT
*
       ORG     0DC00H
PWRON  LDS    #00FFFH
       SEI
       LDX    #IRS
       STX    0FFF8H
       STAA   MPUB
       JSR    INIT
       STAA   BRATE1   SET BIT RATES = 75 BPS
       STAA   BRATE2
       LDAA   #055H     **TEST RAM**
       CLRB
RAMT   LDX    #0FFCH
RAMTST STAA   0,X
       CMPA   0,X
       BNE    RAMERR
       STAA   MPUB
       DEX
       BNE    RAMTST
       TSTB
       BNE    TSTROM
       LDAA   #0AAH     TEST OTHER RAM BITS
       INCB
       BRA    RAMT
TSTROM JSR    ROMTST   **NOP FOR TESTING**
       JSR    CLEAR
       BRA    PWRON1
RAMERR LDAA   #020H     **RAM ERROR**
       STAA   LATCH2   *****
       STAA   MPUB     CRASH
       BRA    RAMERR   *****
***********************************************************
*
*
ROMTST CLRA            **TEST ROM**
       CLRB
       LDX    #PWRON
ROMTSA ADDA   0,X
       ADCB   #0
       STAA   MPUB
       INX
       CPX    #PGMTOP
       BNE    ROMTSA
       CMPA   CKSMA
       BNE    ROMERR
       CMPB   CKSMB
```

```
        BNE    ROMERR
        RTS
*
ROMERR  LDAA   #040H      **ROM ERROR**
        STAA   LATCH2    *****
        STAA   MPUB      CRASH
        BRA    ROMERR    *****
*
PWRON1  JSR    SETUP1
        LDAA   PIA1AD    CLEAR LATCHED INPUTS
        LDAA   PIA1BD
        LDAA   #09DH      ENABLE INTERRUPTS ON ACIA
        STAA   ACIACS
        CLI
        JMP    TDMA
************************************************************
*
*
INIT    LDAA   #01FH      MASTER RESET ACIA
        STAA   ACIACS    DISABLE INTERRUPTS
        LDAA   #0FH
        STAA   LATCH3    RESET I/O FIFOS
        CLRA             I/O PORT & PIA INIT
        STAA   PIA1AC
        STAA   PIA1BC
        STAA   IO1+1
        STAA   IO1+3
        STAA   IO2+1
        STAA   IO2+3
        STAA   IO3+1
        STAA   IO3+3
        STAA   IO4+1
        STAA   IO4+3
        STAA   IO1       SET DATA DIRECTION REG
        STAA   IO2       A - SIDE = INPUTS
        STAA   IO3       FOR I/O PORT
        STAA   IO4
        COMA
        STAA   PIA1AD
        STAA   PIA1BD
        STAA   IO1+2     B-SIDE = OUTPUTS
        STAA   IO2+2     FOR I/O PORT
        STAA   IO3+2
        STAA   IO4+2
        LDAA   #03EH      SET CONTROL REG
        STAA   PIA1AC
        LDAA   #02EH
        STAA   IO1+1
        STAA   IO1+3
        STAA   IO2+1
        STAA   IO2+3
        STAA   IO3+1
        STAA   IO3+3
        STAA   IO4+1
        STAA   IO4+3
        STAA   PIA1BC
        LDAA   PIA1AD    CLEAR LATCHED INPUTS
```

```
       LDAA   PIA1BD
       CLRA
       STAA   PIA1AD
       STAA   LATCH1   CLEAR SIGNAL ACQ. LINES
       STAA   LATCH2
       RTS
**********************************************************
*
*
CLEAR  CLRA            CLEAR RAM MEMORY
       LDX    #0FF8H
CLEARA STAA   0,X
       DEX
       BNE    CLEARA
       STAA   INPT1
       RTS
**********************************************************
*
*
SETUP  BSR    CLEAR
SETUP1 INCA
       STAA   TDMAFG   SETUP RCCOW & CCOW PORTS
       INCA
       STAA   MSDO1
       STAA   MSDO2
       STAA   MSDO3
       STAA   MSDO4
       LDAA   #0FH      INIT FIFO MR INFO
       STAA   MRFTA
       STAA   LATCH3
       LDX    #ERCRX
       STX    RCRXPT
       LDAA   DATA1    CHECK FOR 'TEST 1' MODE
       BITA   #010H
       BEQ    SETUP3
       LDAA   #0FH
       STAA   PTONRX
       LDAA   #024H
       STAA   BRATE1   INIT BIT RATES
       STAA   BRATE2   FOR TEST #1 MODE:
       LDAA   #020H     ALL = 2400 BPS
       STAA   CHRAT1
       STAA   CHRAT2
       STAA   CHRAT3
       STAA   CHRAT4
SETUP3 RTS
**********************************************************
*
*
TDMA   STAA   MPUB     ***BITE OUTPUT***
       LDAA   DATA1    CHECK TDMA SWITCH FLAG
       BITA   #020H     OF T&C PROCESSOR
       BNE    TDMA1
       LDAB   TDMAFG
       BNE    TDMAI
**********************************************************
*
```

```
*NON-TDMA SERVICE
*
        BITA   #01H      CHECK NON-TDMA TX REQ
        BEQ    TDMAA
        LDAA   #01FH     SET TX ACK
        BRA    TDMAB
TDMAA   LDAA   #0FH      CLEAR TX ACK
TDMAB   STAA   LATCH3
        BRA    TDMA
*
TDMAI   SEI
        JSR    INIT     INIT. FOR NON-TDMA
        CLR    TDMAFG
        CLR    PIA1BD
        LDAA   DATA3    RESET TX FIFO
        LDAA   #036H
        STAA   PIA1AC   STROBE "NON-TDMA START"
        LDAA   #03EH
        STAA   PIA1AC
        BRA    TDMA
*
TDMA1   LDAA   TDMAFG
        BNE    RBI
        STAA   PTONRX   TURN ALL PORTS OFF
        BSR    SETUP1   INIT FOR TDMA
        LDAA   #09DH     ENABLE RX INTERRUPTS
        STAA   ACIACS
        CLI
***********************************************************
*
*
RBI     LDAA   PIA1AC   CHECK RCV SETUP RDY
        BMI    RBIA
        JMP    TBI
RBIA    LDAB   SIEN
        BEQ    RBIB
        LDAB   BITELT   SYST INPUT FIFO BITE
        ORAB   #020H     (RX FIFO)
        STAB   BITELT
        CLR    SIEN     CLEAR INPUT FIFO FLAGS
        CLR    SELFTR
        CLR    FILLFG
        CLR    INPTRC
*
*****   BEGIN RECEIVE BURST INIT  *****
*
RBIB    LDAB   PIA1AD   CLR LATCHED STROBE
        LDAA   DATA4    LOAD RCV SETUP DATA
        LDAB   SELFTR
        BEQ    RBIBA
        LDAB   BITELT   *SELF TEST FAIL BITE*
        ORAB   #080H
        STAB   BITELT
RBIBA   ANDA   #0C7H
        TAB
        ANDA   #7
        CLR    MASKIO
```

```
       CMPA   #1
       BEQ    RBI1C
       JMP    RBI2B
*
RBI1C  LDAA   XMTRQ1    **PORT #1 RX BURST INIT**
       BEQ    RBI1CA
       JMP    TBI
RBI1CA LDAA   PTONRX
       ANDA   #1
       BNE    RBI1D
       JMP    TBI
RBI1D  ORAB   CHRAT1
       LDAA   UT1
       BEQ    RBI1E
       LDAA   DATA6     CHECK FIFO'S "O.R."
       BITA   #8
       BNE    RBI1DA
       LDAA   BITELT    ***I/O FIFO #1 BITE***
       ORAA   #2
       STAA   BITELT
RBI1DA CLR    UT1
RBI1E  LDAA   #1
       STAA   MASKIO    INIT I/O ON/OFF MASK
       TSTB
       BMI    RBI1G
       LDAA   MSDO1
       BEQ    RBI1EA
       CLR    FLBFG1    TWO OR MORE MISSING DO'S
       LDAA   #2
       STAA   MSDO1
       JMP    TBI
RBI1EA INCA             **FIRST MISSING DO SETUP**
       STAA   FMSDO1
       STAA   MSDO1
       STAA   FILLFG    SETUP TO STUFF
       LDAA   CHCT1     "FF" INTO BUFFER
       LDAB   CHCT1+1
       ADDB   #32       ADJ COUNT FOR I/O FIFO
       ADCA   #0
       STAA   MDOCT1
       STAB   MDOCT1+1
       LDAB   CHRAT1
       BRA    RBI1H1
RBI1G  LDAA   MSDO1
       CMPA   #2        TWO OR MORE MISSING DO'S?
       BNE    RBI1H
       LDAA   MRFTA
       ORAA   #1        RESET I/O #1 FIFO
       STAA   MRFTA
       STAA   LATCH3
RBI1GA LDX    #SBUFF1    INIT BUFFER #1 AND
       STX    OUTPT1    I/O FIFO #1
       STAA   FLBFG1
       CLRA
       STAA   CHCT1
       LDAA   #0FFH      **RX ELAST BUFF CH**
       TST    BPS16K    16KBPS?
```

```
        BEQ   RBI1GB
        JSR   STUFF     16 KBPS ELASTIC BUF. CHARS.
        STX   INPT1
        LDX   #256      17 MIN. ELASTIC BUFF
        STX   CHCT1
        BRA   RBI1H
RBI1GB  STAA  SBUFF1    RCV BURST
        STAA  SBUFF1+1
        STAA  SBUFF1+2
        STAA  SBUFF1+3
        STAA  SBUFF1+4
        LDAA  #5
        STAA  CHCT1+1
        LDX   #SBUFF1+5
        STX   INPT1
RBI1H   CLR   MSDO1
RBI1H1  LDX   #INPT1    INIT FOR INPUT
        STX   RXPNTR    FIFO SERVICE
        LDAA  BPS16K
        BEQ   RBI1HA    16 KBPS?
        LDX   #EBUFF4
        STX   RXUVAL    INIT 16 KBPS BUFFER PNTRS.
        LDX   #SBUFF1
        STX   RXLVAL
        BRA   RBI1HB
RBI1HA  LDX   #EBUFF1
        STX   RXUVAL
        LDX   #SBUFF1
        STX   RXLVAL
RBI1HB  LDAA  SIGACQ    SET "SIGNAL ACQ. I/O #1"
        ORAA  #1
        STAA  SIGACQ
        STAA  LATCH1
        JMP   RBIS
*
RBI2B   CMPA  #2
        BEQ   RBI2C
        JMP   RBI3B
RBI2C   LDAA  XMTRQ2    **PORT #2 BURST INIT**
        BEQ   RBI2CA
        JMP   TBI
RBI2CA  LDAA  PTONRX
        ANDA  #2
        BNE   RBI2D
        JMP   TBI
RBI2D   ORAB  CHRAT2
        LDAA  UT2
        BEQ   RBI2E
        LDAA  DATA6     CHECK FIFO'S "O.R."
        BITA  #4
        BNE   RBI2DA
        LDAA  BITELT    ***I/O FIFO #2 BITE***
        ORAA  #4
        STAA  BITELT
RBI2DA  CLR   UT2
RBI2E   LDAA  #2
        STAA  MASKIO    INIT I/O ON/OFF MASK
        TSTB
        BMI    RBI2G
        LDAA   MSDO2
        BEQ    RBI2EA
        CLR    FLBFG2   TWO OR MORE MISSING DO'S
        LDAA   #2
        STAA   MSDO2
        JMP    TBI
RBI2EA  INCA            **FIRST MISSING DO SETUP**
        STAA   FMSDO2
        STAA   MSDO2
        STAA   FILLFG   SETUP TO STUFF
        LDAA   CHCT2    "FF" INTO BUFFER
        LDAB   CHCT2+1
        ADDB   #32      ADJ COUNT FOR I/O FIFO
        ADCA   #0
        STAA   MDOCT2
        STAB   MDOCT2+1
        LDAB   CHRAT2
        BRA    RBI2H1
RBI2G   LDAA   MSDO2
        CMPA   #2       TWO OR MORE MISSING DO'S?
        BNE    RBI2H
        LDAA   MRFTA
        ORAA   #2       RESET I/O #2 FIFO
        STAA   MRFTA
        STAA   LATCH3
RBI2GA  LDX    #SBUFF2   INIT BUFFER #2 AND
        STX    OUTPT2   I/O FIFO #2
        STAA   FLBFG2
        CLRA
        STAA   CHCT2
        LDAA   #0FFH     **RX ELAST BUFF CH**
        TST    BPS16K   16KBPS?
        BEQ    RBI2GB
        JSR    STUFF    16KBPS ELASTIC BUF. CHARS.
        STX    INPT2
        LDX    #256     17 MIN ELAST BUFF
        STX    CHCT2
        LDX    #SBUFF1
        STX    OUTPT2
        BRA    RBI2H
RBI2GB  STAA   SBUFF2   RCV BURST
        STAA   SBUFF2+1
        STAA   SBUFF2+2
        STAA   SBUFF2+3
        STAA   SBUFF2+4
        LDAA   #5
        STAA   CHCT2+1
        LDX    #SBUFF2+5
        STX    INPT2
RBI2H   CLR    MSDO2
RBI2H1  LDX    #INPT2   INIT FOR INPUT
        STX    RXPNTR   FIFO SERVICE
        LDAA   BPS16K
        BEQ    RBI2HA   16 KBPS?
        LDX    #EBUFF4
        STX    RXUVAL    INIT. 16KBPS BUF. PNTRS.
        LDX    #SBUFF1
        STX    RXLVAL
        BRA    RBI2HB
RBI2HA  LDX    #EBUFF2
        STX    RXUVAL
        LDX    #SBUFF2
        STX    RXLVAL
RBI2HB  LDAA   SIGACQ    SET "SIGNAL ACQ. I/O #2"
        ORAA   #2
        STAA   SIGACQ
        STAA   LATCH1
        JMP    RBIS
*
RBI3B   CMPA   #3
        BEQ    RBI3C
        JMP    RBI4B
RBI3C   LDAA   XMTRQ3    **PORT #3 BURST INIT**
        BEQ    RBI3CA
        JMP    TBI
RBI3CA  LDAA   PTONRX
        ANDA   #4
        BNE    RBI3D
        JMP    TBI
RBI3D   ORAB   CHRAT3
        LDAA   UT3
        BEQ    RBI3E
        LDAA   DATA6     CHECK FIFO'S "O.R."
        BITA   #2
        BNE    RBI3DA
        LDAA   BITELT    **I/O FIFO #3 BITE***
        ORAA   #8
        STAA   BITELT
RBI3DA  CLR    UT3
RBI3E   LDAA   #4
        STAA   MASKIO    INIT I/O ON/OFF MASK
        TSTB
        BMI    RBI3G
        LDAA   MSDO3
        BEQ    RBI3EA
        CLR    FLBFG3
        LDAA   #2
        STAA   MSDO3
        JMP    TBI
RBI3EA  INCA             **FIRST MISSING DO SETUP**
        STAA   FMSDO3
        STAA   MSDO3
        STAA   FILLFG    SETUP TO STUFF
        LDAA   CHCT3     "FF" INTO BUFFER
        LDAB   CHCT3+1
        ADDB   #32       ADJ COUNT FOR I/O FIFO
        ADCA   #0
        STAA   MDOCT3
        STAB   MDOCT3+1
        LDAB   CHRAT3
        BRA    RBI3H1
RBI3G   LDAA   MSDO3
```

```
        CMPA   #2         TWO OR MORE MISSING DO'S?
        BNE    RBI3H
        LDAA   MRFTA
        ORAA   #4         RESET I/O FIFO #3
        STAA   MRFTA
        STAA   LATCH3
RBI3GA  LDX    #SBUFF3     INIT BUFFER #3 AND
        STX    OUTPT3     I/O FIFO #3
        STAA   FLBFG3
        CLRA
        STAA   CHCT3
        LDAA   #0FFH       **RX ELAST BUFF CH**
        TST    BPS16K     16KBPS?
        BEQ    RBI3GB
        JSR    STUFF      16 KBPS ELASTIC BUF. CHARS.
        STX    INPT3
        LDX    #256       17 MIN ELASTIC BUFF .
        STX    CHCT3
        LDX    #SBUFF1
        STX    OUTPT3
        BRA    RBI3H
RBI3GB  STAA   SBUFF3     RCV BURST
        STAA   SBUFF3+1
        STAA   SBUFF3+2
        STAA   SBUFF3+3
        STAA   SBUFF3+4
        LDAA   #5
        STAA   CHCT3+1
        LDX    #SBUFF3+5
        STX    INPT3
RBI3H   CLR    MSDO3
RBI3H1  LDX    #INPT3     INIT FOR INPUT
        STX    RXPNTR     FIFO SERVICE
        LDAA   BPS16K
        BEQ    RBI3HA
        LDX    #EBUFF4
        STX    RXUVAL     INIT 16KBPS PNTRS
        LDX    #SBUFF1
        STX    RXLVAL
        BRA    RBI3HB
RBI3HA  LDX    #EBUFF3
        STX    RXUVAL
        LDX    #SBUFF3
        STX    RXLVAL
RBI3HB  LDAA   SIGACQ     SET "SIGNAL ACQ. I/O #3"
        ORAA   #4
        STAA   SIGACQ
        STAA   LATCH1
        JMP    RBIS
*
RBI4B   CMPA   #4
        BEQ    RBI4C
        JMP    RBIJ
RBI4C   LDAA   XMTRQ4     **PORT #4 BURST INIT**
        BEQ    RBI4CA
        JMP    TBI
RBI4CA  LDAA   PTONRX
```

```
        ANDA   #8
        BNE    RBI4D
        JMP    TBI
RBI4D   ORAB   CHRAT4
        LDAA   UT4
        BEQ    RBI4E
        LDAA   DATA6     CHECK FIFO'S "O.R."
        BITA   #1
        BNE    RBI4DA
        LDAA   BITELT    **I/O FIFO #4 BITE***
        ORAA   #010H
        STAA   BITELT
RBI4DA  CLR    UT4
RBI4E   LDAA   #8
        STAA   MASKIO    INIT I/O ON/OFF MASK
        TSTB
        BMI    RBI4G
        LDAA   MSDO4
        BEQ    RBI4EA
        CLR    FLBFG4    TWO OR MORE MISSING DO'S
        LDAA   #2
        STAA   MSDO4
        JMP    TBI
RBI4EA  INCA             **FIRST MISSING DO SETUP**
        STAA   FMSDO4
        STAA   MSDO4
        STAA   FILLFG    SETUP TO STUFF
        LDAA   CHCT4     "FF" INTO BUFFER
        LDAB   CHCT4+1
        ADDB   #32       ADJ COUNT FOR I/O FIFO
        ADCA   #0
        STAA   MDOCT4
        STAB   MDOCT4+1
        LDAB   CHRAT4
        BRA    RBI4H1
RBI4G   LDAA   MSDO4
        CMPA   #2        TWO OR MORE MISSING DO'S?
        BNE    RBI4H
        LDAA   MRFTA
        ORAA   #08H       RESET I/O FIFO #4
        STAA   MRFTA
        STAA   LATCH3
RBI4GA  LDX    #SBUFF4    INIT BUFFER #4 AND
        STX    OUTPT4    I/O FIFO #4
        STAA   FLBFG4
        CLRA
        STAA   CHCT4
        LDAA   #0FFH      **RX ELAST BUFF CH**
        TST    BPS16K    16KBPS?
        BEQ    RBI4GB
        JSR    STUFF     16 KBPS ELASTIC BUFF. CHARS
        STX    INPT4
        LDX    #256      17 MIN ELASTIC BUFF
        STX    CHCT4
        LDX    #SBUFF1
        STX    OUTPT4
        BRA    RBI4H
```

```
RBI4GB STAA   SBUFF4   RCV BURST
       STAA   SBUFF4+1
       STAA   SBUFF4+2
       STAA   SBUFF4+3
       STAA   SBUFF4+4
       LDAA   #5
       STAA   CHCT4+1
       LDX    #SBUFF4+5
       STX    INPT4
RBI4H  CLR    MSDO4
RBI4H1 LDX    #INPT4   INIT FOR INPUT
       STX    RXPNTR   FIFO SERVICE
       LDAA   BPS16K
       BEQ    RBI4HA
       LDX    #EBUFF4
       STX    RXUVAL   INIT 16KBPS PNTRS
       LDX    #SBUFF1
       STX    RXLVAL
       BRA    RBI4HB
RBI4HA LDX    #EBUFF4
       STX    RXUVAL
       LDX    #SBUFF4
       STX    RXLVAL
RBI4HB LDAA   SIGACQ   SET "SIGNAL ACQ I/O #4"
       ORAA   #8
       STAA   SIGACQ
       STAA   LATCH1
       JMP    RBIS
***********************************************************
*
*
RBIJ   CMPA   #7       CHECK FOR SELF TEST RX BURS
       BNE    RBIJA
       STAA   SELFTR
       LDAA   #0       INIT. RX FIFO COUNT
       LDAB   #60
       JMP    RBIT
RBIJA  LDX    RCRXPT
       CPX    #ERCRX   R/CCOW RX BUFFER EMPTY?
       BEQ    RBIK
       PSHA            **RCCOW/CCOW ERROR**
       LDAA   BITELT
       ORAA   #1
       STAA   BITELT
       PULA
       LDX    #ERCRX
       STX    RCRXPT
RBIK   LDX    #SRCRX
       CMPA   #6
       BNE    RBIN
       TSTB            CCOW DO?
       BMI    RBIL
       LDAA   #0A5H     OUTPUT CCOW DO
RBIKA  STAA   ACIAD    ABSENT CODE TO SCU
       JMP    TBI
*
RBIL   LDAA   DATA1    TX REQUEST STATUS                  WBR
```

```
        ANDA   #0FH                                        WBR
        ORAA   #0C0H                                       WBR
        STAA   ACIAD    SEND C CODE TO SCP                 WBR
*
        LDAA   #0A1H     CCOW DO PRESENT CODE
RBIM    STAA   0,X
        INX             ADJ R/CCOW RX BUFFER PNTR
        STX    RCRXPT
        STAA   INPTRC
        STAA   SIEN
        JMP    TBI
RBIN    TSTB            RCCOW DO?
        BMI    RBIO
        LDAA   #0A6H     RCCOW DO ABSENT CODE
        BRA    RBIKA
RBIO    LDAA   #0A2H     RCCOW DO PRESENT CODE
        BRA    RBIM
RBIS    BSR    BITRAT
        BRA    RBIT
*************************************************************
*
*
BITRAT TST     BPS16K   TEST 16KBPS
        BNE    RBISF
        ASLB
        BMI    RBIU
        ANDB   #070H
        BNE    RBISA
        LDAB   #13      75 BPS @ 1.386 SEC
        CLRA            13 BYTES
        RTS
RBISA   CMPB   #010H
        BNE    RBISB
        LDAB   #52      300 BPS @ 1.386 SEC
        CLRA            52 BYTES
        RTS
RBISB   CMPB   #020H
        BNE    RBISC
        LDAB   #104     600 BPS @ 1.386 SEC
        CLRA            104 BYTES
        RTS
RBISC   CMPB   #030H
        BNE    RBISD
        LDAB   #208     1200 BPS @ 1.386 SEC
        CLRA            208 BYTES
        RTS
RBISD   CMPB   #040H
        BNE    RBISE
        LDAB   #0A0H     2400 BPS @ 1.386 SEC
        LDAA   #1       416 BYTES
        RTS
RBISE   LDAB   #040H     4800 BPS @ 1.386 SEC
        LDAA   #3       832 BYTES
        RTS
RBISF   LDAB   #0D5H     16000 BPS @ 1.386 SEC
        LDAA   #0AH      2773 BYTES
        RTS
```

```
RBIU    LDAB   #78       75 BPS @ 8.32 SEC
        CLRA             78 BYTES
        RTS
************************************************************
*
*
RBIT    STAA   INCHC     MSB'S OF CH. CNT.
        STAB   INCHC+1    LSB'S OF CH. CNT.
        CLRA
        STAA   INPTRC
        INCA
        STAA   SIEN
************************************************************
*
*
TBI     LDAA   CDRDY     TEST CCOW DATA RDY FLG.
        BEQ    TBI1
        CLR    CDRDY
        LDAA   PA1AD
        ORAA   #020H      SET DATA RDY CCOW*
        BRA    TBI2
TBI1    LDAA   RDRDY     TEST RCCOW RDY FLG.
        BEQ    TBI3
        CLR    RDRDY
        LDAA   PA1AD
        ORAA   #010H      SET DATA RDY RCCOW*
TBI2    STAA   PA1AD
        STAA   PIA1AD
TBI3    LDAA   IRSBIT    IRS BITE?
        BEQ    TBI4
        LDAA   BITELT
        ORAA   #01H
        STAA   BITELT
TBI4    LDAA   PIA1BC    CHECK TX SETUP READY
        BMI    TBIA
        JMP    IO1PT
TBIA    LDAA   SOEN
        BEQ    TBIB
        LDAB   BITELT    SYST OUTPUT FIFO BITE
        ORAB   #040H      (TX FIFO)
        STAB   BITELT
*
***** BEGIN TRANSMIT BURST INITIALIZATION *****
*
TBIB    LDAA   PIA1BD    CLR LATCHED STROBE
        LDAA   DATA3     LOAD TX SETUP DATA
        LDAB   DATA5     CHECK TX INHIBIT
        BITB   #1
        BEQ    TIBA
        ANDA   #7
        CMPA   #7
        BEQ    TBIBA
        JMP    IO1PT     TX INHIBIT: ABORT
TBIBA   JMP    TBIG      ALLOW SELF TEST BURST
TIBA    ANDA   #047H
        TAB
        ANDA   #7
```

```
       CMPA   #1
       BNE    TBI2B
       LDAA   PTONRX     IS PORT ON?
       BITA   #1
       BNE    TBI1BA
       JMP    IO1PT      ABORT:  PORT IS OFF
*
TBI1BA TST    BPS16K
       BEQ    TBI1E
       LDX    #SBUFF1    16 KBPS
       STX    TXLVAL
       LDX    #EBUFF4
       STX    TXUVAL
       BRA    TBI1EA
TBI1E  LDX    #SBUFF1    INIT FOR OUTPUT
       STX    TXLVAL   FIFO SERVICE
       LDX    #EBUFF1
       STX    TXUVAL
TBI1EA LDX    #INPT1
       STX    TXPNTR
       CLRA
       STAA   UT1
       STAA   MDOCT1
       STAA   MDOCT1+1
       STAA   FMSDO1
       LDAA   #2
       STAA   MSDO1
       LDAA   #0FEH
       STAA   XMASK
       LDAA   #0EFH
       STAA   MASKR
       LDAA   GROUP1
       BEQ    TBI1F
       STAA   GRP1FG
TBI1F  CLRA
       STAA   GROUP1
       ORAB   CHRAT1
       JMP    TBIN
*
TBI2B  CMPA   #2
       BNE    TBI3B
       LDAA   PTONRX     IS PORT ON?
       BITA   #2
       BNE    TBI2BA
       JMP    IO1PT      ABORT:  PORT IS OFF
TBI2BA TST    BPS16K
       BEQ    TBI2E
       LDX    #SBUFF1    16 KBPS
       STX    TXLVAL
       LDX    #EBUFF4
       STX    TXUVAL
       BRA    TBI2EA
TBI2E  LDX    #SBUFF2    INIT. FOR OUTPUT
       STX    TXLVAL   FIFO SERVICE
       LDX    #EBUFF2
       STX    TXUVAL
TBI2EA LDX    #INPT2
```

```
        STX    TXPNTR
        CLRA
        STAA   UT2
        STAA   MDOCT2
        STAA   MDOCT2+1
        STAA   FMSDO2
        LDAA   #2
        STAA   MSDO2
        LDAA   #0FDH
        STAA   XMASK
        LDAA   #0DFH
        STAA   MASKR
        LDAA   GROUP2
        BEQ    TBI2F
        STAA   GRP1FG
TBI2F   CLRA
        STAA   GROUP2
        ORAB   CHRAT2
        JMP    TBIN
*
TBI3B   CMPA   #3
        BNE    TBI4B
        LDAA   PTONRX     IS PORT ON?
        BITA   #4
        BNE    TBI3BA
        JMP    IO1PT      ABORT:  PORT IS OFF
TBI3BA  TST    BPS16K
        BEQ    TBI3E
        LDX    #SBUFF1    16 KBPS
        STX    TXLVAL
        LDX    #EBUFF4
        STX    TXUVAL
        BRA    TBI3EA
TBI3E   LDX    #SBUFF3    INIT. FOR OUPTUT
        STX    TXLVAL    FIFO SERVICE
        LDX    #EBUFF3
        STX    TXUVAL
TBI3EA  LDX    #INPT3
        STX    TXPNTR
        CLRA
        STAA   UT3
        STAA   MDOCT3
        STAA   MDOCT3+1
        STAA   FMSDO3
        LDAA   #2
        STAA   MSDO3
        LDAA   #0FBH
        STAA   XMASK
        LDAA   #0BFH
        STAA   MASKR
        LDAA   GROUP3
        BEQ    TBI3F
        STAA   GRP1FG
TBI3F   CLRA
        STAA   GROUP3
        ORAB   CHRAT3
        JMP    TBIN
```

```
*
TBI4B   CMPA   #4
        BNE    TBIG
        LDAA   PTONRX     IS PORT ON?
        BITA   #8
        BNE    TBI4BA
        JMP    IO1PT      ABORT:  PORT IS OFF
TBI4BA  TST    BPS16K
        BEQ    TBI4E
        LDX    #SBUFF1    16 KBPS
        STX    TXLVAL
        LDX    #EBUFF4
        STX    TXUVAL
        BRA    TBI4EA
TBI4E   LDX    #SBUFF4    INIT. FOR OUTPUT
        STX    TXLVAL    FIFO SERVICE
        LDX    #EBUFF4
        STX    TXUVAL
TBI4EA  LDX    #INPT4
        STX    TXPNTR
        CLRA
        STAA   UT4
        STAA   MDOCT4
        STAA   MDOCT4+1
        STAA   FMSDO4
        LDAA   #2
        STAA   MSDO4
        LDAA   #0F7H
        STAA   XMASK
        LDAA   #07FH
        STAA   MASKR
        LDAA   GROUP4
        BEQ    TBI4F
        STAA   GRP1FG
TBI4F   CLRA
        STAA   GROUP4
        ORAB   CHRAT4
        JMP    TBIN
*
TBIG    CMPA   #7         CHECK FOR SELF TEST TX BURS
        BNE    TBIGA
        STAA   SOEN
        STAA   SELFTT
        LDX    #08000H    DUMMY VALUE FOR TXPNTR
        STX    TXPNTR
        LDX    #28        SELF TEST CH CNT - 32
        STX    OUTCHC
        LDAA   #32
        LDAB   #033H       SELF TEST TX CHAR
TBIG1   STAB   PIA1BD
        DECA              OUTPUT 32 SELF TEST
        BNE    TBIG1      CHARS. TO TX FIFO
        JMP    RBI
TBIGA   CMPA   #6
        BNE    TBIJ
        LDAA   PA1AD      IF NO DATA READY, OUTPUT
        BITA   #020H       13 BYTES OF 11001100
        BNE    TBIGC
        LDAA   #0CCH      **NO CCOW DATA RDY**
        LDX    #13
TBIGB   STAA   PIA1BD
        DEX
        BNE    TBIGB
        BRA    TBIHB
TBIGC   LDX    #SCTXB
        STX    CTXPT
TBIH    LDAA   0,X
        STAA   PIA1BD   PUT CHAR IN OUTPUT FIFO
        INX
        CPX    #ECTXB
        BNE    TBIH
        LDAA   PA1AD
        ANDA   #0DFH      CLR DATA READY CCOW
TBIHA   STAA   PIA1AD
        STAA   PA1AD
*
*
TBIHB   ASLB
        BMI    TBIHC
        JMP    IO1PT
TBIHC   CLRA
        STAA   PIA1BD
        STAA   PIA1BD
        STAA   PIA1BD
        JMP    IO1PT
TBIJ    LDX    #SRTXB
        STX    RTXPT
TBIK    LDAA   0,X
        STAA   PIA1BD   PUT CHAR IN OUTPUT FIFO
        INX
        CPX    #ERTXB
        BNE    TBIK
        LDAA   PA1AD
        ANDA   #0EFH      CLR DATA READY RCCOW
        BRA    TBIHA
TBIN    JSR    BITRAT   GET TX CH. CNT.
TBIO    STAA   OUTCHC   MSB'S OF CH. CNT.
        STAB   OUTCHC+1   LSB'S OF CH. CNT.
        CLRA
        INCA
        STAA   SOEN
TBIQ    LDX    TXPNTR
        LDAA   GRP1FG   CHECK IF FIRST GROUP
        BEQ    TBIS
TBIQ1   LDAA   OUTCHC   ***COMPUTE FILL COUNT***
        LDAB   OUTCHC+1
        ADDB   #6       ADD ELASTIC BUFF(48 BITS)
        ADCA   #0
        SUBB   5,X      BURST CH CNT + ELASTIC BUFF
        SBCA   4,X      - BUFF CH CNT = FILL COUNT
        BCS    TBIR
        BNE    TBIRA
        TSTB
        BNE    TBIRA
```

```
TBIR   CLR    GRP1FG
       BRA    TBIS      NO FILL REQUIRED
TBIRA  STAA   FILLCT
       STAB   FILLCT+1
       LDAA   4,X       NUMBER OF GOOD CHARS IN FIR
       LDAB   5,X       BURST=BUFF CH CNT - ELASTIC
       SUBB   #6        SUBT ELASTIC BUFF(48 BITS)
       SBCA   #0
       BCC    TBIRB
       CLRA             ERROR: ELASTIC BUFFER NOT
       STAA   SOEN      YET FULL FOR TRANSMISSION
       LDAA   BITELT
       ORAA   #040H      **TX FIFO BITE LITE ON**
       STAA   BITELT
       BRA    IO1PT
TBIRB  STAA   OUTCHC
       STAB   OUTCHC+1
TBIS   JSR    SOFAA     *SERVICE OUTPUT FIFO*
******************************************************
*
*
*
IO1PT  LDAA   PTONRX    PORT #1 ON?
       ANDA   #1
       BEQ    IO1A1
       CLR    P1OFF
       JMP    IO1A
IO1A1  LDAA   P1OFF     PORT IS OFF
       BEQ    IO1A2
       JMP    IO2PT
IO1A2  STAA   CHCT1
       STAA   CHCT1+1
       STAA   XMTRQ1
       STAA   UT1
       LDX    #0
       STX    LGDPT1
       STAA   OUFG1
       STAA   EMPTY1
       LDAA   #2
       STAA   MSDO1
       INCA
       STAA   FLBFG1
       LDAA   MRFTA
       ORAA   #1        RESET FIFO I/O #1
       ANDA   #0EFH      CLR "XMT ACK"
       STAA   MRFTA
       STAA   LATCH3
       STAA   P1OFF
       LDAA   PA1AD
       ANDA   #0FEH      CLEAR "DATA RDY PORT #1"
       STAA   PA1AD
       STAA   PIA1AD
       LDX    #SBUFF1
       STX    INPT1
       STX    OUTPT1
       LDAA   SIGACQ    CLR "SIGNAL ACQ. I/O #1
       ANDA   #0FEH
```

```
         STAA    SIGACQ
         STAA    LATCH1
         LDX     TXPNTR
         CPX     #INPT1
         BNE     IO1A3
         CLR     SOEN
IO1A3    JMP     IO2PT
***********************************************************
*
*   INPUT SERVICE SUBROUTINE
*
IO1LS    LDAA    #1
         STAA    XFER1
         LDAA    IO1       OBTAIN CHAR
         LDX     INPT1     GET PUT PNTR
         STAA    0,X       PUT IT
         INX               INCR PUT PNTR
         CPX     OUTPT1    TEST OVERFLOW
         BNE     IO1LSX    BRANCH NO OVERFLOW
         LDAA    MRFTA     **OVERFLOW**
         ANDA    #0EFH      CLR "XMT ACK"
         ORAA    #1        RESET FIFO
         STAA    MRFTA
         STAA    LATCH3
         STAA    OUFG1     SET OUFG FLAG
         INS               *REMOVE RETURN
         INS               *
         JMP     IO2PT
IO1LSX   LDAA    BPS16K
         BEQ     IO1LSA
         CPX     #EBUFF4    16 KBPS
         BNE     IO1LSB
         LDX     #SBUFF1
         BRA     IO1LSB
IO1LSA   CPX     #EBUFF1    TEST END OF INPUT
         BNE     IO1LSB    NO
         LDX     #SBUFF1    GET INITIAL VALUE
IO1LSB   STX     INPT1
         LDX     CHCT1     GET CHAR COUNT
         INX               INCR COUNT
         STX     CHCT1     PUT BACK
         RTS
***********************************************************
*
*LOAD I/O #1 FIFO WITH 1 CHAR. ROUTINE
*
IO1LF    LDX     OUTPT1
         LDAA    RXINHB                                        WBR
         BITA    #1        IS THE EXTEND TRANSMIT FLAG SET?    WBR
         BNE     IO1LF1    YES:RX NOT ALLOWED                  WBR
         LDAA    0,X
         STAA    IO1+2     OUTPUT CH. TO I/O #1 FIFO
IO1LF1   INX
         LDAB    BPS16K
         BEQ     IO1LFA
         CPX     #EBUFF4    16 KBPS
         BNE     IO1LFB
```

```
        LDX    #SBUFF1
        BRA    IO1LFB
IO1LFA  CPX    #EBUFF1
        BNE    IO1LFB
        LDX    #SBUFF1
IO1LFB  STX    OUTPT1
        LDAA   #1
        STAA   XFER1
        LDAA   FMSDO1    TEST MISSING DO
        BEQ    IO1LFC
        LDX    MDOCT1
        DEX
        STX    MDOCT1
        BNE    IO1LFC
        LDAA   SIGACQ
        ANDA   #0FEH
        STAA   SIGACQ
        STAA   LATCH1
        CLR    FMSDO1
IO1LFC  LDX    CHCT1     ADJUST CHAR COUNT
        DEX
        STX    CHCT1
        BNE    IO1LFD
        INS              REMOVE RETURN
        INS
        JMP    IO1YE1    GO CHECK UNDERFLOW
IO1LFD  RTS
*********************************************************
*
*LOAD I/O #1 FIFO WITH 3 BUF. CH. ROUTINE
*
LD1     LDX    OUTPT1
        LDAA   0,X
        STAA   IO1+2
        LDAA   1,X
        STAA   IO1+2
        LDAA   2,X
        STAA   IO1+2
        INX
        INX
        INX
        STX    OUTPT1
        LDX    CHCT1
        DEX
        DEX
        DEX
        STX    CHCT1
        RTS
*********************************************************
*
*****  BEGIN SERVICING I/O PORT #1   *****
*
IO1A    LDAA   PTONRX    RX ONLY?
        BITA   #010H
        BEQ    IO1AA
        JMP    IO1Y
IO1AA   LDAA   DATA1
```

```
         BITA   #1
         BNE    IO1B
         JMP    IO1Y
IO1B     ASLA
         BPL    IO1C
         LDAA   XMTRQ1
         BNE    IO1C
         LDAA   EMPTY1
         BNE    IO1CA
         LDX    CHCT1       R/T SELECTED: CHECK RCV DON
         BEQ    IO1BB
IO1BA    JMP    IO1YD
IO1BB    LDAA   DATA6       TEST: I/O FIFO UNDERFLOW
         BITA   #8
         BNE    IO1BA       BRANCH: NO UNDERFLOW
IO1C     LDAA   XMTRQ1
         BNE    IO1D
         LDAA   EMPTY1
         BEQ    IO1E
IO1CA    CLR    EMPTY1      GET HERE W/MULTIPLE
         STAA   XMTRQ1      TX REQ PER FRAME
         LDX    #0
         STX    LGDPT1
         LDAA   MRFTA
         ORAA   #010H        SET "TX ACK"
         STAA   MRFTA
         STAA   LATCH3
IO1D     LDAA   OUFG1
         BEQ    IO1H
         STAA   XFER1
         JMP    IO2PT
IO1E     LDAA   DATA5       CHECK TX INHIBIT
         ANDA   #1
         BEQ    IO1F
         JMP    IO2PT
IO1F     LDAA   #1          **INIT FOR PORT #1**
         STAA   XMTRQ1      **DATA TRANSMISSION**
         STAA   GROUP1
         STAA   XFER1
         STAA   FLBFG1
         LDAA   SIGACQ
         ORAA   #010H        SELECT I/O INPUT MODE
         ANDA   #0FEH        CLR "SIG. ACQ. #1"
         STAA   SIGACQ
         STAA   LATCH1
         LDAA   MRFTA
         ORAA   #1          TOGGLE FIFO'S RESET
         STAA   LATCH3
         CLR    IO1+2       CLR PIA B-SIDE
         ANDA   #0FEH
         ORAA   #010H        SET "XMT ACK" I/O #1
         STAA   MRFTA
         STAA   LATCH3
         LDX    #SBUFF1+6
         STX    INPT1
         LDX    #SBUFF1
         STX    OUTPT1
```

```
        LDAA   #0FFH     STUFF 6 ELASTIC
        STAA   0,X      BUFFER CHARACTERS (CC)
        STAA   1,X
        STAA   2,X
        STAA   3,X
        STAA   4,X
        STAA   5,X
        LDX    #6
        STX    CHCT1
        JMP    IO2PT
IO1G    LDX    #SBUFF1
        STX    INPT1
        STX    OUTPT1
        LDX    #0
        STX    CHCT1
        LDAA   SIGACQ   CLR "SIGNAL ACQ. I/O #1
        ANDA   #0FEH
        STAA   SIGACQ
        STAA   LATCH1
        JMP    IO2PT
IO1H    LDAA   DATA5    CHECK TX INHIBIT
        ANDA   #1
        BEQ    IO1HA
        JMP    IO1YA1
IO1HA   LDAB   FLBFG1
        BEQ    IO1HB
        JMP    IO1O
IO1HB   LDAA   DATA6    TEST FIFO "O.R."
        BITA   #8
        BNE    IO1I
        JMP    IO1XP
IO1I    JSR    IO1LS    LOAD & STORE 1 CHAR
        LDAA   DATA6
        BITA   #8
        BNE    IO1K
        JMP    IO1XP
IO1K    JSR    IO1LS    LOAD & STORE 2 CHAR
        LDAA   DATA6
        BITA   #8
        BNE    IO1M
        JMP    IO1XP
IO1M    JSR    IO1LS    LOAD & STORE 3 CHAR
        LDAA   DATA6
        BITA   #8
        BNE    IO1N2
        JMP    IO1XP
IO1N2   JSR    IO1LS    LOAD & STORE 4 CHAR
        LDAA   BPS16K
        BNE    IO1N3
        JMP    IO1XP
IO1N3   LDAA   DATA6    *16KBPS:  ALLOW MORE*
        BITA   #8       *I/O SERVICING*
        BNE    IO1N4
        JMP    IO1XP
IO1N4   JSR    IO1LS    LOAD & STORE 5 CHAR.
        LDAA   DATA6
        BITA   #8
```

```
              BNE     IO1N5
              JMP     IO1XP
       IO1N5  JSR     IO1LS     LOAD & STORE 6 CHAR.
              LDAA    DATA6
              BITA    #8
              BNE     IO1N6
              JMP     IO1XP
       IO1N6  JSR     IO1LS     LOAD & STORE 7 CHAR.
              LDAA    DATA6
              BITA    #8
              BNE     IO1N7
              JMP     IO1XP
       IO1N7  JSR     IO1LS     LOAD & STORE 8 CHAR.
              LDAA    DATA6
              BITA    #8
              BNE     IO1N8
              JMP     IO1XP
       IO1N8  JSR     IO1LS     LOAD & STORE 9 CHAR.
              LDAA    DATA6
              BITA    #8
              BNE     IO1N9
              JMP     IO1XP
       IO1N9  JSR     IO1LS     LOAD & STORE 10 CHAR.
              LDAA    DATA6
              BITA    #8
              BNE     IO1N10
              JMP     IO1XP
       IO1N10 JSR     IO1LS     LOAD & STORE 11 CHAR.
              JMP     IO1XP
       IO1O   LDAA    DATA6
              BITA    #8
              BEQ     IO1XP
       IO1P   CLR     FLBFG1
              LDAA    PA1AD     SET "DATA RDY PORT #1"
              ORAA    #1
              STAA    PA1AD
              STAA    PIA1AD
              JMP     IO1I
       IO1XP  LDAA    EMPTY1    **TEST IF SETUP**
              BPL     IO1XQ     **LAST GOOD DATA PNTR**
              LDX     INPT1     DO SETUP
              STX     LGDPT1
              LDAA    #1
              STAA    EMPTY1
       IO1XQ  JMP     IO2PT
       IO1Y   LDAA    XMTRQ1
              BEQ     IO1YB
              LDAA    MRFTA
              ANDA    #0EFH      CLEAR "TX ACK"
              STAA    MRFTA
              STAA    LATCH3
              CLR     XMTRQ1    FIRST TIME THAT
              LDAA    #0FFH      TX REQ = FALSE
              STAA    EMPTY1
              LDAA    PA1AD
              ANDA    #1
              BNE     IO1YA
```

```
        STAA   CHCT1       NO DATA:  ABORT INPUT
        STAA   CHCT1+1
        STAA   EMPTY1
        INCA
        STAA   FLBFG1
        LDAA   MRFTA
        ORAA   #1
        STAA   MRFTA
        STAA   LATCH3
        JMP    IO2PT
IO1YA   JMP    IO1D
IO1YB   LDAA   EMPTY1
        BNE    IO1YA
        JMP    IO1YD
IO1YA1  CLRA
        STAA   XMTRQ1
        STAA   EMPTY1
        LDAA   PA1AD
        ANDA   #0FEH
        STAA   PIA1AD    CLEAR "DATA RDY PORT #1"
        STAA   PA1AD
        LDAA   MRFTA
        ORAA   #1        RESET I/O #1 FIFO
        ANDA   #0EFH      CLR TX ACK I/O #1
        STAA   MRFTA
        STAA   LATCH3
IO1YC   LDAA   MRFTA     **INIT I/O PORT **
        ORAA   #1        **TO OUTPUT**
        STAA   MRFTA     **RX BURST DATA**
        STAA   LATCH3    RESET I/O #1 FIFO
        LDAA   SIGACQ
        ANDA   #0EFH      SELECT I/O OUTPUT MODE
        STAA   SIGACQ
        STAA   LATCH1
IO1YC1  CLR    FMSDO1
        LDAA   #2
        STAA   MSDO1
        STAA   FLBFG1
        JMP    IO1G
IO1YD   LDAA   FLBFG1
        BEQ    IO1YF
        STAA   XFER1
        LDAA   CHCT1+1
        CMPA   #10       RCV ELASTIC BUFFER FULL?
        BGE    IO1YE
        JMP    IO2PT
IO1YE   CLR    FLBFG1
        STAA   UT1
        LDAA   SIGACQ    SET OUTPUT MODE
        ANDA   #0EFH
        STAA   SIGACQ
        STAA   LATCH1
        LDAA   MRFTA
        ANDA   #0FEH      CLR RESET ON I/O FIFO
        STAA   MRFTA
        STAA   LATCH3
        JSR    LD1
        JSR    LD1      LOAD FIFO W/6 BUF. CH.
        LDAA   BPS16K
        BEQ    IO1YE0
        JSR    LD1      16KBPS SO:
        JSR    LD1
        JSR    LD1
        JSR    LD1
        JSR    LD1      LOAD FIFO WITH 21 BUF. CH.
IO1YE0  JMP    IO2PT
IO1YE1  LDAA   #1
        STAA   XFER1
        LDAA   UT1
        BNE    IO1YE0
        LDAA   DATA6    CHECK FIFO UNDERFLOW
        BITA   #8       BY TESTING FIFO "O.R."
        BEQ    IO1YC1   **UNDERFLOW**
        JMP    IO2PT
IO1YF   LDX    CHCT1
        BEQ    IO1YE1
        LDAA   DATA6
        LDAB   UT1
        BNE    IO1YFA
        BITA   #8
        BEQ    IO1YC1
IO1YFA  BITA   #080H     CHECK FIFO "I.R.":  FULL?
        BEQ    IO2PT
IO1YG   JSR    IO1LF    OUTPUT 1 CH. TO FIFO
        LDAA   DATA6
        BITA   #080H     CHECK "I.R.": FULL?
        BEQ    IO2PT
IO1YH   JSR    IO1LF    OUTPUT 2 CH. TO FIFO
        LDAA   DATA6
        BITA   #080H     CHECK "I.R.": FULL?
        BEQ    IO2PT
IO1YI   JSR    IO1LF    OUTPUT 3 CH. TO FIFO
        LDAA   BPS16K
        BEQ    IO2PT
IO1YJ   LDAA   DATA6    16KBPS: SERVICE 10 CHARS.
        BITA   #080H     CHECK "I.R.": FULL?
        BEQ    IO2PT
IO1YK   JSR    IO1LF    OUTPUT 4 CH. TO FIFO
        LDAA   DATA6
        BITA   #080H     CHECK "I.R.": FULL?
        BEQ    IO2PT
IO1YL   JSR    IO1LF    OUTPUT 5 CH. TO FIFO
        LDAA   DATA6
        BITA   #080H     CHECK "I.R.": FULL?
        BEQ    IO2PT
IO1YM   JSR    IO1LF    OUTPUT 6 CH. TO FIFO
        LDAA   DATA6
        BITA   #080H     CHECK "I.R.": FULL?
        BEQ    IO2PT
IO1YN   JSR    IO1LF    OUTPUT 7 CH. TO FIFO
        LDAA   DATA6
        BITA   #080H     CHECK "I.R.": FULL?
        BEQ    IO2PT
IO1YO   JSR    IO1LF    OUTPUT 8 CH. TO FIFO
```

```
        LDAA   DATA6
        BITA   #080H      CHECK "I.R.": FULL?
        BEQ    IO2PT
IO1YP   JSR    IO1LF     OUTPUT 9 CH. TO FIFO
        LDAA   DATA6
        BITA   #080H      CHECK "I.R.": FULL?
        BEQ    IO2PT
IO1YQ   JSR    IO1LF     OUTPUT 10 CH. TO FIFO
***********************************************************
*
*
*
IO2PT   LDAA   PTONRX    PORT #2 ON?
        ANDA   #2
        BEQ    IO2A1
        CLR    P2OFF
        JMP    IO2A
IO2A1   LDAA   P2OFF     PORT IS OFF
        BEQ    IO2A2
        JMP    IO3PT
IO2A2   STAA   CHCT2
        STAA   CHCT2+1
        STAA   XMTRQ2
        STAA   UT2
        LDX    #0
        STX    LGDPT2
        STAA   OUFG2
        STAA   EMPTY2
        LDAA   #2
        STAA   MSDO2
        INCA
        STAA   FLBFG2
        LDAA   MRFTA
        ORAA   #2        RESET FIFO I/O #2
        ANDA   #0DFH
        STAA   MRFTA
        STAA   LATCH3
        STAA   P2OFF
        LDAA   PA1AD
        ANDA   #0FDH      CLEAR "DATA RDY PORT #2"
        STAA   PA1AD
        STAA   PIA1AD
        LDX    #SBUFF2
        STX    INPT2
        STX    OUTPT2
        LDAA   SIGACQ    CLR "SIGNAL ACQ. I/O #2"
        ANDA   #0FDH
        STAA   SIGACQ
        STAA   LATCH1
        LDX    TXPNTR
        CPX    #INPT2
        BNE    IO2A3
        CLR    SOEN
IO2A3   JMP    IO3PT
***********************************************************
*
*   INPUT SERVICE ROUTINE
```

```
*
IO2LS   LDAA   #1
        STAA   XFER2
        LDAA   IO2       OBTAIN CHAR
        LDX    INPT2     GET PUT PNTR
        STAA   0,X       PUT IT
        INX              INCR PUT PNTR
        CPX    OUTPT2    TEST OVERFLOW
        BNE    IO2LSX    BRANCH NO OVERFLOW
        LDAA   MRFTA     **OVERFLOW**
        ANDA   #0DFH      CLR "XMT ACK"
        ORAA   #2        RESET FIFO
        STAA   MRFTA
        STAA   LATCH3
        STAA   OUFG2     SET OUFG FLAG
        INS              *REMOVE RETURN
        INS              *
        JMP    IO3PT
IO2LSX  LDAA   BPS16K
        BEQ    IO2LSA
        CPX    #EBUFF4    16 KBPS
        BNE    IO2LSB
        LDX    #SBUFF1
        BRA    IO2LSB
IO2LSA  CPX    #EBUFF2    TEST END OF INPUT
        BNE    IO2LSB    NO
        LDX    #SBUFF2    GET INITIAL VALUE
IO2LSB  STX    INPT2
        LDX    CHCT2     GET CHAR COUNT
        INX              INCR COUNT
        STX    CHCT2     PUT BACK
        RTS
************************************************************
*
*LOAD I/O #2 FIFO WITH 1 CHAR ROUTINE
*
IO2LF   LDX    OUTPT2
        LDAA   RXINHB                                      WBR
        BITA   #2        IS THE EXTEND TRANSMIT FLAG SET?  WBR
        BNE    IO2LF1    YES:RX NOT ALLOWED                WBR
        LDAA   0,X
        STAA   IO2+2     OUTPUT CH. TO I/O #2 FIFO
IO2LF1  INX              ADJ. OUTPUT PNTR.
        LDAB   BPS16K
        BEQ    IO2LFA
        CPX    #EBUFF4    16 KBPS
        BNE    IO2LFB
        LDX    #SBUFF1
        BRA    IO2LFB
IO2LFA  CPX    #EBUFF2
        BNE    IO2LFB
        LDX    #SBUFF2
IO2LFB  STX    OUTPT2
        LDAA   #1
        STAA   XFER2
        LDAA   FMSDO2    TEST MISSING DO
        BEQ    IO2LFC
```

```
       LDX    MDOCT2
       DEX
       STX    MDOCT2
       BNE    IO2LFC
       LDAA   SIGACQ
       ANDA   #0FDH
       STAA   SIGACQ
       STAA   LATCH1
       CLR    FMSDO2
IO2LFC LDX    CHCT2     ADJUST CHAR COUNT
       DEX
       STX    CHCT2
       BNE    IO2LFD
       INS              REMOVE RETURN
       INS
       JMP    IO2YE1    GO CHECK UNDERFLOW
IO2LFD RTS
*************************************************************
*
* LOAD FIFO I/O #2 WITH 3 BUF. CH. ROUTINE
*
LD2    LDX    OUTPT2
       LDAA   0,X
       STAA   IO2+2
       LDAA   1,X
       STAA   IO2+2
       LDAA   2,X
       STAA   IO2+2
       INX
       INX
       INX
       STX    OUTPT2
       LDX    CHCT2
       DEX
       DEX
       DEX
       STX    CHCT2
       RTS
*************************************************************
*
*****  BEGIN SERVICING I/O PORT #2   *****
*
IO2A   LDAA   PTONRX    RX ONLY?
       BITA   #020H
       BEQ    IO2AA
       JMP    IO2Y
IO2AA  LDAA   DATA1
       BITA   #2
       BNE    IO2B
       JMP    IO2Y
IO2B   ASLA
       BPL    IO2C
       LDAA   XMTRQ2
       BNE    IO2C
       LDAA   EMPTY2
       BNE    IO2CA
       LDX    CHCT2     R/T SELECTED: CHECK RCV DON
```

```
        BEQ    IO2BB
IO2BA   JMP    IO2YD
IO2BB   LDAA   DATA6     TEST: I/O FIFO UNDERFLOW
        BITA   #4
        BNE    IO2BA     BRANCH: NO UNDERFLOW
IO2C    LDAA   XMTRQ2
        BNE    IO2D
        LDAA   EMPTY2
        BEQ    IO2E
IO2CA   CLR    EMPTY2    GET HERE W/MULTPLE
        STAA   XMTRQ2    TX REQ PER FRAME
        LDX    #0
        STX    LGDPT2
        LDAA   MRFTA
        ORAA   #020H      SET "TX ACK"
        STAA   MRFTA
        STAA   LATCH3
IO2D    LDAA   OUFG2
        BNE    IO2DA
        JMP    IO2H
IO2DA   STAA   XFER2
        JMP    IO3PT
IO2E    LDAA   DATA5     CHECK TX INHIBIT
        ANDA   #1
        BEQ    IO2F
        JMP    IO3PT
IO2F    LDAA   #1        **INIT FOR PORT #2**
        STAA   XMTRQ2    **DATA TRANSMISSION**
        STAA   GROUP2
        STAA   XFER2
        STAA   FLBFG2
        LDAA   SIGACQ
        ORAA   #020H      SELECT I/O INPUT MODE
        ANDA   #0FDH      CLR "SIG. ACQ. #2"
        STAA   SIGACQ
        STAA   LATCH1
        LDAA   MRFTA
        ORAA   #2        TOGGLE FIFO'S RESET
        STAA   LATCH3
        CLR    IO2+2     CLR PIA B-SIDE
        ANDA   #0FDH
        ORAA   #020H      SET "XMT ACK" I/O #2
        STAA   MRFTA
        STAA   LATCH3
        LDAA   BPS16K
        BEQ    IO2FA
        LDX    #SBUFF1+6   16KBPS
        STX    INPT2
        LDX    #SBUFF1
        BRA    IO2FB
IO2FA   LDX    #SBUFF2+6
        STX    INPT2
        LDX    #SBUFF2
IO2FB   STX    OUTPT2
        LDAA   #0FFH      STUFF 6 ELASTIC
        STAA   0,X      BUFFER CHARACTERS (CC)
        STAA   1,X
```

```
         STAA   2,X
         STAA   3,X
         STAA   4,X
         STAA   5,X
         LDX    #6
         STX    CHCT2
         JMP    IO3PT
IO2G     LDAA   BPS16K
         BEQ    IO2GA
         LDX    #SBUFF1   16KBPS
         BRA    IO2GB
IO2GA    LDX    #SBUFF2
IO2GB    STX    INPT2
         STX    OUTPT2
         LDX    #0
         STX    CHCT2
         LDAA   SIGACQ   CLR "SIGNAL ACQ. I/O #2"
         ANDA   #0FDH
         STAA   SIGACQ
         STAA   LATCH1
         JMP    IO3PT
IO2H     LDAA   DATA5    CHECK TX INHIBIT
         ANDA   #1
         BEQ    IO2HA
         JMP    IO2YA1
IO2HA    LDAB   FLBFG2
         BEQ    IO2HB
         JMP    IO2O
IO2HB    LDAA   DATA6    TEST FIFO "O.R."
         BITA   #4
         BNE    IO2I
         JMP    IO2XP
IO2I     JSR    IO2LS    LOAD & STORE 1 CHAR
         LDAA   DATA6
         BITA   #4
         BNE    IO2K
         JMP    IO2XP
IO2K     JSR    IO2LS    LOAD & STORE 2 CHAR
         LDAA   DATA6
         BITA   #4
         BNE    IO2M
         JMP    IO2XP
IO2M     JSR    IO2LS    LOAD & STORE 3 CHAR
         LDAA   DATA6
         BITA   #4
         BNE    IO2N2
         JMP    IO2XP
IO2N2    JSR    IO2LS    LOAD & STORE 4 CHAR
         LDAA   BPS16K
         BNE    IO2N3
         JMP    IO2XP
IO2N3    LDAA   DATA6    *16KBPS:  ALLOW MORE*
         BITA   #4       *I/O SERVICING*
         BNE    IO2N4
         JMP    IO2XP
IO2N4    JSR    IO2LS    LOAD & STORE 5 CHAR.
         LDAA   DATA6
```

```
        BITA  #4
        BNE   IO2N5
        JMP   IO2XP
IO2N5   JSR   IO2LS     LOAD & STORE 6 CHAR.
        LDAA  DATA6
        BITA  #4
        BNE   IO2N6
        JMP   IO2XP
IO2N6   JSR   IO2LS     LOAD & STORE 7 CHAR.
        LDAA  DATA6
        BITA  #4
        BNE   IO2N7
        JMP   IO2XP
IO2N7   JSR   IO2LS     LOAD & STORE 8 CHAR.
        LDAA  DATA6
        BITA  #4
        BNE   IO2N8
        JMP   IO2XP
IO2N8   JSR   IO2LS     LOAD & STORE 9 CHAR.
        LDAA  DATA6
        BITA  #4
        BNE   IO2N9
        JMP   IO2XP
IO2N9   JSR   IO2LS     LOAD & STORE 10 CHAR.
        LDAA  DATA6
        BITA  #4
        BNE   IO2N10
        JMP   IO2XP
IO2N10  JSR   IO2LS     LOAD & STORE 11 CHAR.
        JMP   IO2XP
IO2O    LDAA  DATA6
        BITA  #4
        BEQ   IO2XP
IO2P    CLR   FLBFG2
        LDAA  PA1AD     SET "DATA RDY PORT #2"
        ORAA  #2
        STAA  PA1AD
        STAA  PIA1AD
        JMP   IO2I
IO2XP   LDAA  EMPTY2    **TEST IF SETUP**
        BPL   IO2XQ     **LAST GOOD DATA PNTR**
        LDX   INPT2     DO SETUP
        STX   LGDPT2
        LDAA  #1
        STAA  EMPTY2
IO2XQ   JMP   IO3PT
IO2Y    LDAA  XMTRQ2
        BEQ   IO2YB
        LDAA  MRFTA
        ANDA  #0DFH      CLR "TX ACK"
        STAA  MRFTA
        STAA  LATCH3
        CLR   XMTRQ2    FIRST TIME THAT
        LDAA  #0FFH      TX REQ = FALSE
        STAA  EMPTY2
        LDAA  PA1AD
        ANDA  #2
        BNE    IO2YA
        STAA   CHCT2        NO DATA:  ABORT INPUT
        STAA   CHCT2+1
        STAA   EMPTY2
        INCA
        STAA   FLBFG2
        LDAA   MRFTA
        ORAA   #2
        STAA   MRFTA
        STAA   LATCH3
        JMP    IO3PT
IO2YA   JMP    IO2D
IO2YB   LDAA   EMPTY2
        BNE    IO2YA
        JMP    IO2YD
IO2YA1  CLRA
        STAA   XMTRQ2
        STAA   EMPTY2
        LDAA   PA1AD
        ANDA   #0FDH
        STAA   PIA1AD     CLEAR "DATA RDY PORT #2"
        STAA   PA1AD
        LDAA   MRFTA
        ORAA   #2         RESET I/O #2 FIFO
        ANDA   #0DFH       CLR TX ACK I/O #2
        STAA   MRFTA
        STAA   LATCH3
IO2YC   LDAA   MRFTA      **INIT I/O PORT**
        ORAA   #2         **TO OUTPUT**
        STAA   MRFTA      **RX BURST DATA**
        STAA   LATCH3     RESET I/O #2 FIFO
        LDAA   SIGACQ
        ANDA   #0DFH       SELECT I/O OUTPUT MODE
        STAA   SIGACQ
        STAA   LATCH1
IO2YC1  CLR    FMSDO2
        LDAA   #2
        STAA   MSDO2
        STAA   FLBFG2
        JMP    IO2G
IO2YD   LDAA   FLBFG2
        BEQ    IO2YF
        STAA   XFER2
        LDAA   CHCT2+1
        CMPA   #10        RCV ELASTIC BUFFER FULL?
        BGE    IO2YE
        JMP    IO3PT
IO2YE   CLR    FLBFG2
        STAA   UT2
        LDAA   SIGACQ     SET OUTPUT MODE
        ANDA   #0DFH
        STAA   SIGACQ
        STAA   LATCH1
        LDAA   MRFTA
        ANDA   #0FDH       CLR REST ON I/O FIFO
        STAA   MRFTA
        STAA   LATCH3
        JSR    LD2
        JSR    LD2      LOAD FIFO W/6 BUF. CH.
        LDAA   BPS16K
        BEQ    IO2YE0
        JSR    LD2      16 KBPS SO:
        JSR    LD2
        JSR    LD2
        JSR    LD2
        JSR    LD2      LOAD FIFO W/21 BUF. CH.
IO2YE0  JMP    IO3PT
IO2YE1  LDAA   #1
        STAA   XFER2
        LDAA   UT2
        BNE    IO2YE0
        LDAA   DATA6    CHECK FIFO UNDERFLOW
        BITA   #4       BY TESTING FIFO "O.R."
        BEQ    IO2YC1   **UNDERFLOW**
        JMP    IO3PT
IO2YF   LDX    CHCT2
        BEQ    IO2YE1
        LDAA   DATA6
        LDAB   UT2
        BNE    IO2YFA
        BITA   #4
        BEQ    IO2YC1
IO2YFA  BITA   #040H     CHECK FIFO "I.R.":  FULL?
        BEQ    IO3PT
IO2YG   JSR    IO2LF    OUTPUT 1 CH. TO FIFO
        LDAA   DATA6
        BITA   #040H     CHECK "I.R.": FULL?
        BEQ    IO3PT
IO2YH   JSR    IO2LF    OUTPUT 2 CH. TO FIFO
        LDAA   DATA6
        BITA   #040H     CHECK "I.R.":  FULL?
        BEQ    IO3PT
IO2YI   JSR    IO2LF    OUTPUT 3 CH. TO FIFO
        LDAA   BPS16K
        BEQ    IO3PT
IO2YJ   LDAA   DATA6    16 KBPS:  SERVICE 10 CHARS.
        BITA   #040H     CHECK "I.R.": FULL?
        BEQ    IO3PT
IO2YK   JSR    IO2LF    OUTPUT 4 CH. TO FIFO
        LDAA   DATA6
        BITA   #040H     CHECK "I.R.": FULL?
        BEQ    IO3PT
IO2YL   JSR    IO2LF    OUTPUT 5 CH. TO FIFO
        LDAA   DATA6
        BITA   #040H     CHECK "I.R.": FULL?
        BEQ    IO3PT
IO2YM   JSR    IO2LF    OUTPUT 6 CH. TO FIFO
        LDAA   DATA6
        BITA   #040H     CHECK "I.R.": FULL?
        BEQ    IO3PT
IO2YN   JSR    IO2LF    OUTPUT 7 CH. TO FIFO
        LDAA   DATA6
        BITA   #040H     CHECK "I.R.": FULL?
        BEQ    IO3PT
```

```
IO2YO  JSR    IO2LF    OUTPUT 8 CH. TO FIFO
       LDAA   DATA6
       BITA   #040H     CHECK "I.R.": FULL?
       BEQ    IO3PT
IO2YP  JSR    IO2LF    OUTPUT 9 CH. TO FIFO
       LDAA   DATA6
       BITA   #040H     CHECK "I.R.": FULL?
       BEQ    IO3PT
IO2YQ  JSR    IO2LF    OUTPUT 10 CH. TO FIFO
*************************************************************
*
*
*
IO3PT  LDAA   PTONRX   PORT #3 ON?
       ANDA   #4
       BEQ    IO3A1
       CLR    P3OFF
       JMP    IO3A
IO3A1  LDAA   P3OFF    PORT IS OFF
       BEQ    IO3A2
       JMP    IO4PT
IO3A2  STAA   CHCT3
       STAA   CHCT3+1
       STAA   XMTRQ3
       STAA   UT3
       LDX    #0
       STX    LGDPT3
       STAA   OUFG3
       STAA   EMPTY3
       LDAA   #2
       STAA   MSDO3
       INCA
       STAA   FLBFG3
       LDAA   MRFTA
       ORAA   #4       RESET FIFO I/O #3
       ANDA   #0BFH     CLR XMT ACK I/O #3
       STAA   MRFTA
       STAA   LATCH3
       STAA   P3OFF
       LDAA   PA1AD
       ANDA   #0FBH     CLEAR "DATA RDY PORT #3"
       STAA   PA1AD
       STAA   PIA1AD
       LDX    #SBUFF3
       STX    INPT3
       STX    OUTPT3
       LDAA   SIGACQ   CLR "SIGNAL ACQ. I/O #3"
       ANDA   #0FBH
       STAA   SIGACQ
       STAA   LATCH1
       LDX    TXPNTR
       CPX    #INPT3
       BNE    IO3A3
       CLR    SOEN
IO3A3  JMP    IO4PT
*************************************************************
*
```

```
*   INPUT SERVICE SUBROUTINE
*
IO3LS   LDAA    #1
        STAA    XFER3
        LDAA    IO3       OBTAIN CHAR
        LDX     INPT3     GET PUT PNTR
        STAA    0,X       PUT IT
        INX               INCR PUT PNTR
        CPX     OUTPT3    TEST OVERFLOW
        BNE     IO3LSX    BRANCH NO OVERFLOW
        LDAA    MRFTA     **OVERFLOW**
        ANDA    #0BFH      CLR XMT ACK I/O #3
        ORAA    #4        RESET FIFO I/O #3
        STAA    MRFTA
        STAA    LATCH3
        STAA    OUFG3     SET OUFG FLAG
        INS               *REMOVE RETURN
        INS               *
        JMP     IO4PT
IO3LSX  LDAA    BPS16K
        BEQ     IO3LSA
        CPX     #EBUFF4    16 KBPS
        BNE     IO3LSB
        LDX     #SBUFF1
        BRA     IO3LSB
IO3LSA  CPX     #EBUFF3    TEST END OF INPUT
        BNE     IO3LSB    NO
        LDX     #SBUFF3    GET INITIAL VALUE
IO3LSB  STX     INPT3
        LDX     CHCT3     GET CHAR. COUNT
        INX               INCR COUNT
        STX     CHCT3     PUT BACK
        RTS
************************************************************
*
*LOAD I/O #3 FIFO WITH 1 CHAR. ROUTINE
*
IO3LF   LDX     OUTPT3
        LDAA    RXINHB                                       WBR
        BITA    #4        IS THE EXTEND TRANSMIT FLAG SET?   WBR
        BNE     IO3LF1    YES:RX NOT ALLOWED                 WBR
        LDAA    0,X
        STAA    IO3+2     OUTPUT CH. TO I/O #3 FIFO
IO3LF1  INX               ADJ. OUTPUT PNTR.
        LDAB    BPS16K
        BEQ     IO3LFA
        CPX     #EBUFF4    16KBPS
        BNE     IO3LFB
        LDX     #SBUFF1
        BRA     IO3LFB
IO3LFA  CPX     #EBUFF3
        BNE     IO3LFB
        LDX     #SBUFF3
IO3LFB  STX     OUTPT3
        LDAA    #1
        STAA    XFER3
        LDAA    FMSDO3    TEST MISSING DO
        BEQ     IO3LFC
        LDX     MDOCT3
        DEX
        STX     MDOCT3
        BNE     IO3LFC
        LDAA    SIGACQ
        ANDA    #0FBH
        STAA    SIGACQ
        STAA    LATCH1
        CLR     FMSDO3
IO3LFC  LDX     CHCT3     ADJ CHAR. COUNT
        DEX
        STX     CHCT3
        BNE     IO3LFD
        INS               REMOVE RETURN
        INS
        JMP     IO3YE1    GO CHECK UNDERFLOW
IO3LFD  RTS
***********************************************************
*
*LOAD FIFO I/O #3 WITH 3 BUF. CH. ROUTINE
*
LD3     LDX     OUTPT3
        LDAA    0,X
        STAA    IO3+2
        LDAA    1,X
        STAA    IO3+2
        LDAA    2,X
        STAA    IO3+2
        INX
        INX
        INX
        STX     OUTPT3
        LDX     CHCT3
        DEX
        DEX
        DEX
        STX     CHCT3
        RTS
***********************************************************
*
*****   BEGIN SERVICING I/O PORT #3    *****
*
IO3A    LDAA    PTONRX    RX ONLY?
        BITA    #040H
        BEQ     IO3AA
        JMP     IO3Y
IO3AA   LDAA    DATA1
        BITA    #4
        BNE     IO3B
        JMP     IO3Y
IO3B    ASLA
        BPL     IO3C
        LDAA    XMTRQ3
        BNE     IO3C
        LDAA    EMPTY3
        BNE     IO3CA
```

```
         LDX    CHCT3     R/T SELECTED: CHECK RCV DON
         BEQ    IO3BB
IO3BA    JMP    IO3YD
IO3BB    LDAA   DATA6     TEST: I/O FIFO UNDERFLOW
         BITA   #2
         BNE    IO3BA     BRANCH:  NO UNDERFLOW
IO3C     LDAA   XMTRQ3
         BNE    IO3D
         LDAA   EMPTY3
         BEQ    IO3E
IO3CA    CLR    EMPTY3    GET HERE W/MULTIPLE
         STAA   XMTRQ3    TX REQ PER FRAME
         LDX    #0
         STX    LGDPT3
         LDAA   MRFTA
         ORAA   #040H      SET "TX ACK"
         STAA   MRFTA
         STAA   LATCH3
IO3D     LDAA   OUFG3
         BNE    IO3DA
         JMP    IO3H
IO3DA    STAA   XFER3
         JMP    IO4PT
IO3E     LDAA   DATA5     CHECK TX INHIBIT
         ANDA   #1
         BEQ    IO3F
         JMP    IO4PT
IO3F     LDAA   #1        **INIT FOR PORT #3**
         STAA   XMTRQ3    **DATA TRANSMISSION**
         STAA   GROUP3
         STAA   XFER3
         STAA   FLBFG3
         LDAA   SIGACQ
         ORAA   #040H      SET I/O INPUT MODE
         ANDA   #0FBH      CLR "SIG. ACQ. #3"
         STAA   SIGACQ
         STAA   LATCH1
         LDAA   MRFTA
         ORAA   #4        TOGGLE FIFO'S RESET
         STAA   LATCH3
         CLR    IO3+2     CLR PIA-B SIDE
         ANDA   #0FBH
         ORAA   #040H      SET XMT ACK I/O #3
         STAA   MRFTA
         STAA   LATCH3
         LDAA   BPS16K
         BEQ    IO3FA
         LDX    #SBUFF1+6   16KBPS
         STX    INPT3
         LDX    #SBUFF1
         BRA    IO3FB
IO3FA    LDX    #SBUFF3+6
         STX    INPT3
         LDX    #SBUFF3
IO3FB    STX    OUTPT3
         LDAA   #0FFH      STUFF 6 ELASTIC
         STAA   0,X       BUFFER CHARACTERS (CC)
```

```
        STAA   1,X
        STAA   2,X
        STAA   3,X
        STAA   4,X
        STAA   5,X
        LDX    #6
        STX    CHCT3
        JMP    IO4PT
IO3G    LDAA   BPS16K
        BEQ    IO3GA
        LDX    #SBUFF1   16KBPS
        BRA    IO3GB
IO3GA   LDX    #SBUFF3
IO3GB   STX    INPT3
        STX    OUTPT3
        LDX    #0
        STX    CHCT3
        LDAA   SIGACQ    CLR "SIGNAL ACQ. I/O #3"
        ANDA   #0FBH
        STAA   SIGACQ
        STAA   LATCH1
        JMP    IO4PT
IO3H    LDAA   DATA5     CHECK TX INHIBIT
        ANDA   #1
        BEQ    IO3HA
        JMP    IO3YA1
IO3HA   LDAB   FLBFG3
        BEQ    IO3HB
        JMP    IO3O
IO3HB   LDAA   DATA6     TEST FIFO "O.R."
        BITA   #2
        BNE    IO3I
        JMP    IO3XP
IO3I    JSR    IO3LS     LOAD & STORE 1 CHAR
        LDAA   DATA6
        BITA   #2
        BNE    IO3K
        JMP    IO3XP
IO3K    JSR    IO3LS     LOAD & STORE 2 CHAR
        LDAA   DATA6
        BITA   #2
        BNE    IO3M
        JMP    IO3XP
IO3M    JSR    IO3LS     LOAD & STORE 3 CHAR
        LDAA   DATA6
        BITA   #2
        BNE    IO3N2
        JMP    IO3XP
IO3N2   JSR    IO3LS     LOAD & STORE 4 CHAR
        LDAA   BPS16K
        BNE    IO3N3
        JMP    IO3XP
IO3N3   LDAA   DATA6     *16KBPS:  ALLOW MORE*
        BITA   #2        *I/O SERVICING*
        BNE    IO3N4
        JMP    IO3XP
IO3N4   JSR    IO3LS     LOAD & STORE 5 CHAR.
```

```
        LDAA   DATA6
        BITA   #2
        BNE    IO3N5
        JMP    IO3XP
IO3N5   JSR    IO3LS    LOAD & STORE 6 CHAR.
        LDAA   DATA6
        BITA   #2
        BNE    IO3N6
        JMP    IO3XP
IO3N6   JSR    IO3LS    LOD & STORE 7 CHAR.
        LDAA   DATA6
        BITA   #2
        BNE    IO3N7
        JMP    IO3XP
IO3N7   JSR    IO3LS    LOAD & STORE 8 CHAR.
        LDAA   DATA6
        BITA   #2
        BNE    IO3N8
        JMP    IO3XP
IO3N8   JSR    IO3LS    LOAD & STORE 9 CHAR.
        LDAA   DATA6
        BITA   #2
        BNE    IO3N9
        JMP    IO3XP
IO3N9   JSR    IO3LS    LOAD & STORE 10 CHAR.
        LDAA   DATA6
        BITA   #2
        BNE    IO3N10
        JMP    IO3XP
IO3N10  JSR    IO3LS    LOAD & STORE 11 CHAR.
        JMP    IO3XP
IO3O    LDAA   DATA6
        BITA   #2
        BEQ    IO3XP
IO3P    CLR    FLBFG3
        LDAA   PA1AD    SET "DATA RDY PORT #3"
        ORAA   #4
        STAA   PA1AD
        STAA   PIA1AD
        JMP    IO3I
IO3XP   LDAA   EMPTY3   **TEST IF SETUP**
        BPL    IO3XQ    **LAST GOOD DATA PNTR**
        LDX    INPT3    DO SETUP
        STX    LGDPT3
        LDAA   #1
        STAA   EMPTY3
IO3XQ   JMP    IO4PT
IO3Y    LDAA   XMTRQ3
        BEQ    IO3YB
        LDAA   MRFTA
        ANDA   #0BFH     CLR "TX ACK"
        STAA   MRFTA
        STAA   LATCH3
        CLR    XMTRQ3   FIRST TIME THAT
        LDAA   #0FFH     TX REQ = FALSE
        STAA   EMPTY3
        LDAA   PA1AD
```

```
               ANDA    #4
               BNE     IO3YA
               STAA    CHCT3       NO DATA:  ABORT INPUT
               STAA    CHCT3+1
               STAA    EMPTY3
               INCA
               STAA    FLBFG3
               LDAA    MRFTA
               ORAA    #4
               STAA    MRFTA
               STAA    LATCH3
               JMP     IO4PT
      IO3YA    JMP     IO3D
      IO3YB    LDAA    EMPTY3
               BNE     IO3YA
               JMP     IO3YD
      IO3YA1   CLRA
               STAA    XMTRQ3
               STAA    EMPTY3
               LDAA    PA1AD
               ANDA    #0FBH
               STAA    PIA1AD    CLEAR "DATA RDY PORT #3"
               STAA    PA1AD
               LDAA    MRFTA
               ORAA    #4        RESET I/O #1 FIFO
               ANDA    #0BFH      CLR TX ACK I/O #1
               STAA    MRFTA
               STAA    LATCH3
      IO3YC    LDAA    MRFTA     **INIT I/O PORT**
               ORAA    #4        **TO OUTPUT**
               STAA    MRFTA     **RX BURST DATA**
               STAA    LATCH3    RESET I/O #3 FIFO
               LDAA    SIGACQ
               ANDA    #0BFH      SELECT I/O OUTPUT MODE
               STAA    SIGACQ
               STAA    LATCH1
      IO3YC1   CLR     FMSDO3
               LDAA    #2
               STAA    MSDO3
               STAA    FLBFG3
               JMP     IO3G
      IO3YD    LDAA    FLBFG3
               BEQ     IO3YF
               STAA    XFER3
               LDAA    CHCT3+1
               CMPA    #10       RCV ELASTIC BUFFER FULL?
               BGE     IO3YE
               JMP     IO4PT
      IO3YE    CLR     FLBFG3
               STAA    UT3
               LDAA    SIGACQ    SET OUTPUT MODE
               ANDA    #0BFH
               STAA    SIGACQ
               STAA    LATCH1
               LDAA    MRFTA
               ANDA    #0FBH      CLR RESET ON I/O FIFO
               STAA    MRFTA
```

```
        STAA  LATCH3
        JSR   LD3
        JSR   LD3       LOAD FIFO W/6 BUF. CH.
        LDAA  BPS16K
        BEQ   IO3YE0
        JSR   LD3
        JSR   LD3
        JSR   LD3
        JSR   LD3
        JSR   LD3       LOAD FIFO W/21 BUF. CH.
IO3YE0  JMP   IO4PT
IO3YE1  LDAA  #1
        STAA  XFER3
        LDAA  UT3
        BNE   IO3YE0
        LDAA  DATA6     CHECK FIFO UNDERFLOW
        BITA  #2        BY TESTING FIFO "O.R."
        BEQ   IO3YC1    **UNDERFLOW**
        JMP   IO4PT
IO3YF   LDX   CHCT3
        BEQ   IO3YE1
        LDAA  DATA6
        LDAB  UT3
        BNE   IO3YFA
        BITA  #2
        BEQ   IO3YC1
IO3YFA  BITA  #020H      CHECK "I.R.":  FULL?
        BEQ   IO4PT
IO3YG   JSR   IO3LF     OUTPUT 1 CH. TO FIFO
        LDAA  DATA6
        BITA  #020H      CHECK "I.R.":  FULL?
        BEQ   IO4PT
IO3YH   JSR   IO3LF     OUTPUT 2 CH. TO FIFO
        LDAA  DATA6
        BITA  #020H      CHECK "I.R.": FULL?
        BEQ   IO4PT
IO3YI   JSR   IO3LF     OUTPUT 3 CH. TO FIFO
        LDAA  BPS16K
        BEQ   IO4PT
IO3YJ   LDAA  DATA6     16 KBPS:  SERVICE 10 CHARS.
        BITA  #020H      CHECK "I.R.": FULL?
        BEQ   IO4PT
IO3YK   JSR   IO3LF     OUTPUT 4 CH. TO FIFO
        LDAA  DATA6
        BITA  #020H      CHECK "I.R.": FULL?
        BEQ   IO4PT
IO3YL   JSR   IO3LF     OUTPUT 5 CH. TO FIFO
        LDAA  DATA6
        BITA  #020H      CHECK "I.R.": FULL?
        BEQ   IO4PT
IO3YM   JSR   IO3LF     OUTPUT 6 CH. TO FIFO
        LDAA  DATA6
        BITA  #020H      CHECK "I.R.": FULL?
        BEQ   IO4PT
IO3YN   JSR   IO3LF     OUTPUT 7 CH. TO FIFO
        LDAA  DATA6
        BITA  #020H      CHECK "I.R.": FULL?
```

```
         BEQ    IO4PT
IO3YO    JSR    IO3LF     OUTPUT 8 CH. TO FIFO
         LDAA   DATA6
         BITA   #020H      CHECK "I.R.": FULL?
         BEQ    IO4PT
IO3YP    JSR    IO3LF     OUTPUT 9 CH. TO FIFO
         LDAA   DATA6
         BITA   #020H      CHECK "I.R.": FULL?
         BEQ    IO4PT
IO3YQ    JSR    IO3LF     OUTPUT 10 CH. TO FIFO
********************************************************
*
*
*
IO4PT    LDAA   PTONRX    PORT #4 ON?
         ANDA   #8
         BEQ    IO4A1
         CLR    P4OFF
         JMP    IO4A
IO4A1    LDAA   P4OFF     PORT IS OFF
         BEQ    IO4A2
         JMP    BSERV
IO4A2    STAA   CHCT4
         STAA   CHCT4+1
         STAA   XMTRQ4
         STAA   UT4
         LDX    #0
         STX    LGDPT4
         STAA   OUFG4
         STAA   EMPTY4
         LDAA   #2
         STAA   MSDO4
         INCA
         STAA   FLBFG4
         LDAA   MRFTA
         ORAA   #8        RESET FIFO I/O #4
         ANDA   #07FH      CLR XMT ACK
         STAA   MRFTA
         STAA   LATCH3
         STAA   P4OFF
         LDAA   PA1AD
         ANDA   #0F7H      CLEAR "DATA RDY PORT #4"
         STAA   PA1AD
         STAA   PIA1AD
         LDX    #SBUFF4
         STX    INPT4
         STX    OUTPT4
         LDAA   SIGACQ    CLR "SIGNAL ACQ. I/O #4"
         ANDA   #0F7H
         STAA   SIGACQ
         STAA   LATCH1
         LDX    TXPNTR
         CPX    #INPT4
         BNE    IO4A3
         CLR    SOEN
IO4A3    JMP    BSERV
********************************************************
```

```
*
*   INPUT SERVICE SUBROUTINE
*
IO4LS  LDAA   #1
       STAA   XFER4
       LDAA   IO4      OBTAIN CHAR
       LDX    INPT4    GET PUT PNTR
       STAA   0,X      PUT IT
       INX             INCR PUT PNTR
       CPX    OUTPT4   TEST OVERFLOW
       BNE    IO4LSX   BRANCH NO OVERFLOW
       LDAA   MRFTA    **OVERFLOW**
       ANDA   #07FH     CLR XMT ACK
       ORAA   #8       RESET FIFO
       STAA   MRFTA
       STAA   LATCH3
       STAA   OUFG4    SET OUFG FLAG
       INS             *REMOVE RETURN
       INS             *
       JMP    BSERV
IO4LSX LDAA   BPS16K
       BEQ    IO4LSA
       CPX    #EBUFF4   16 KBPS
       BNE    IO4LSB
       LDX    #SBUFF1
       BRA    IO4LSB
IO4LSA CPX    #EBUFF4   TEST END OF INPUT
       BNE    IO4LSB   NO
       LDX    #SBUFF4   GET INITIAL VALUE
IO4LSB STX    INPT4
       LDX    CHCT4    GET CHAR COUNT
       INX             INCR COUNT
       STX    CHCT4    PUT BACK
       RTS
***********************************************************
*
*LOAD I/O #4 FIFO WITH 1 CHAR. ROUTINE
*
IO4LF  LDX    OUTPT4
       LDAA   RXINHB                                       WBR
       BITA   #8       IS THE EXTEND TRANSMIT FLAG SET?    WBR
       BNE    IO4LF1   YES:RX NOT ALLOWED                  WBR
       LDAA   0,X
       STAA   IO4+2    OUTPUT CH. TO I/O #4 FIFO
IO4LF1 INX             ADJ. OUTPUT PNTR.
       LDAB   BPS16K
       BEQ    IO4LFA
       CPX    #EBUFF4   16KBPS
       BNE    IO4LFB
       LDX    #SBUFF1
       BRA    IO4LFB
IO4LFA CPX    #EBUFF4
       BNE    IO4LFB
       LDX    #SBUFF4
IO4LFB STX    OUTPT4
       LDAA   #1
       STAA   XFER4
```

```
        LDAA   FMSDO4   TEST MISSING DO
        BEQ    IO4LFC
        LDX    MDOCT4
        DEX
        STX    MDOCT4
        BNE    IO4LFC
        LDAA   SIGACQ
        ANDA   #0F7H
        STAA   SIGACQ
        STAA   LATCH1
        CLR    FMSDO4
IO4LFC  LDX    CHCT4    ADJUST CHR COUNT
        DEX
        STX    CHCT4
        BNE    IO4LFD
        INS             REMOVE RETURN
        INS
        JMP    IO4YE1   GO CHECK UNDERFLOW
IO4LFD  RTS
***********************************************************
*
*LOAD FIFO I/O #4 WITH 3 BUF. CH. ROUTINE
*
LD4     LDX    OUTPT4
        LDAA   0,X
        STAA   IO4+2
        LDAA   1,X
        STAA   IO4+2
        LDAA   2,X
        STAA   IO4+2
        INX
        INX
        INX
        STX    OUTPT4
        LDX    CHCT4
        DEX
        DEX
        DEX
        STX    CHCT4
        RTS
***********************************************************
*
*****   BEGIN SERVICING I/O PORT #4    *****
*
IO4A    LDAA   PTONRX   RX ONLY?
        BITA   #080H
        BEQ    IO4AA
        JMP    IO4Y
IO4AA   LDAA   DATA1
        BITA   #8
        BNE    IO4B
        JMP    IO4Y
IO4B    ASLA
        BPL    IO4C
        LDAA   XMTRQ4
        BNE    IO4C
        LDAA   EMPTY4
```

```
           BNE    IO4CA
           LDX    CHCT4     R/T SELECTED: CHECK RCV DON
           BEQ    IO4BB
     IO4BA JMP    IO4YD
     IO4BB LDAA   DATA6     TEST: I/O FIFO UNDERFLOW
           BITA   #1
           BNE    IO4BA     BRANCH: NO UNDERFLOW
     IO4C  LDAA   XMTRQ4
           BNE    IO4D
           LDAA   EMPTY4
           BEQ    IO4E
     IO4CA CLR    EMPTY4    GET HERE W/MULTIPLE
           STAA   XMTRQ4    TX REQ PER FRAME
           LDX    #0
           STX    LGDPT4
           LDAA   MRFTA
           ORAA   #080H      SET "TX ACK"
           STAA   MRFTA
           STAA   LATCH3
     IO4D  LDAA   OUFG4
           BNE    IO4DA
           JMP    IO4H
     IO4DA STAA   XFER4
           JMP    BSERV
     IO4E  LDAA   DATA5     CHECK TX INHIBIT
           ANDA   #1
           BEQ    IO4F
           JMP    BSERV
     IO4F  LDAA   #1        **INIT FOR PORT #4**
           STAA   XMTRQ4    **DATA TRANSMISSION**
           STAA   GROUP4
           STAA   XFER4
           STAA   FLBFG4
           LDAA   SIGACQ
           ORAA   #080H      SELECT I/O INPUT MODE
           ANDA   #0F7H      CLR "SIG. ACQ. #4"
           STAA   SIGACQ
           STAA   LATCH1
           LDAA   MRFTA
           ORAA   #8        TOGGLE FIFO'S RESET
           STAA   LATCH3
           CLR    IO4+2     CLR PIA-B SIDE
           ANDA   #0F7H
           ORAA   #080H      SET XMT ACK I/O #4
           STAA   MRFTA
           STAA   LATCH3
           LDAA   BPS16K
           BEQ    IO4FA
           LDX    #SBUFF1+6   16KBPS
           STX    INPT4
           LDX    #SBUFF1
           BRA    IO4FB
     IO4FA LDX    #SBUFF4+6
           STX    INPT4
           LDX    #SBUFF4
     IO4FB STX    OUTPT4
           LDAA   #0FFH      STUFF 6 ELASTIC
```

```
        STAA   0,X       BUFFER CHARACTERS (CC)
        STAA   1,X
        STAA   2,X
        STAA   3,X
        STAA   4,X
        STAA   5,X
        LDX    #6
        STX    CHCT4
        JMP    BSERV
IO4G    LDAA   BPS16K
        BEQ    IO4GA
        LDX    #SBUFF1    16 KBPS
        BRA    IO4GB
IO4GA   LDX    #SBUFF4
IO4GB   STX    INPT4
        STX    OUTPT4
        LDX    #0
        STX    CHCT4
        LDAA   SIGACQ    CLR "SIGNAL ACQ. I/O #4"
        ANDA   #0F7H
        STAA   SIGACQ
        STAA   LATCH1
        JMP    BSERV
IO4H    LDAA   DATA5     CHECK TX INHIBIT
        ANDA   #1
        BEQ    IO4HA
        JMP    IO4YA1
IO4HA   LDAB   FLBFG4
        BEQ    IO4HB
        JMP    IO4O
IO4HB   LDAA   DATA6     TEST FIFO "O.R."
        BITA   #1
        BNE    IO4I
        JMP    IO4XP
IO4I    JSR    IO4LS     LOAD & STORE 1 CHAR
        LDAA   DATA6
        BITA   #1
        BNE    IO4K
        JMP    IO4XP
IO4K    JSR    IO4LS     LOAD & STORE 2 CHAR
        LDAA   DATA6
        BITA   #1
        BNE    IO4M
        JMP    IO4XP
IO4M    JSR    IO4LS     LOAD $ STORE 3 CHAR.
        LDAA   DATA6
        BITA   #1
        BNE    IO4N2
        JMP    IO4XP
IO4N2   JSR    IO4LS     LOAD & STORE 4 CHAR
        LDAA   BPS16K
        BNE    IO4N3
        JMP    IO4XP
IO4N3   LDAA   DATA6     *16 KBPS:  ALLOW MORE*
        BITA   #1        *I/O SERVICING*
        BNE    IO4N4
        JMP    IO4XP
```

```
IO4N4   JSR     IO4LS      LOAD & STORE 5 CHAR.
        LDAA    DATA6
        BITA    #1
        BNE     IO4N5
        JMP     IO4XP
IO4N5   JSR     IO4LS      LOAD & STORE 6 CHAR.
        LDAA    DATA6
        BITA    #1
        BNE     IO4N6
        JMP     IO4XP
IO4N6   JSR     IO4LS      LOAD & STORE 7 CHAR.
        LDAA    DATA6
        BITA    #1
        BNE     IO4N7
        JMP     IO4XP
IO4N7   JSR     IO4LS      LOAD & STORE 8 CHAR.
        LDAA    DATA6
        BITA    #1
        BNE     IO4N8
        JMP     IO4XP
IO4N8   JSR     IO4LS      LOAD & STORE 9 CHAR.
        LDAA    DATA6
        BITA    #1
        BNE     IO4N9
        JMP     IO4XP
IO4N9   JSR     IO4LS      LOAD & STORE 10 CHAR.
        LDAA    DATA6
        BITA    #1
        BNE     IO4N10
        JMP     IO4XP
IO4N10 JSR      IO4LS      LOAD & STORE 11 CHAR.
        JMP     IO4XP
IO4O    LDAA    DATA6
        BITA    #1
        BEQ     IO4XP
IO4P    CLR     FLBFG4
        LDAA    PA1AD      SET "DATA RDY PORT #4"
        ORAA    #8
        STAA    PA1AD
        STAA    PIA1AD
        JMP     IO4I
IO4XP   LDAA    EMPTY4     **TEST IF SETUP**
        BPL     IO4XQ      **LAST GOOD DATA PNTR**
        LDX     INPT4      DO SETUP
        STX     LGDPT4
        LDAA    #1
        STAA    EMPTY4
IO4XQ   JMP     BSERV
IO4Y    LDAA    XMTRQ4
        BEQ     IO4YB
        LDAA    MRFTA
        ANDA    #07FH       CLR "TX ACK"
        STAA    MRFTA
        STAA    LATCH3
        CLR     XMTRQ4     FIRST TIME THAT
        LDAA    #0FFH       TX REQ = FALSE
        STAA    EMPTY4
        LDAA   PA1AD
        ANDA   #8
        BNE    IO4YA
        STAA   CHCT4      NO DATA:  ABORT INPUT
        STAA   CHCT4+1
        STAA   EMPTY4
        INCA
        STAA   FLBFG4
        LDAA   MRFTA
        ORAA   #8
        STAA   MRFTA
        STAA   LATCH3
        JMP    BSERV
IO4YA   JMP    IO4D
IO4YB   LDAA   EMPTY4
        BNE    IO4YA
        JMP    IO4YD
IO4YA1  CLRA
        STAA   XMTRQ4
        STAA   EMPTY4
        LDAA   PA1AD
        ANDA   #0F7H
        STAA   PIA1AD   CLEAR "DATA RDY PORT #4"
        STAA   PA1AD
        LDAA   MRFTA
        ORAA   #8       RESET I/O #4 FIFO
        ANDA   #07FH     CLR TX ACK I/O #4
        STAA   MRFTA
        STAA   LATCH3
IO4YC   LDAA   MRFTA    **INIT I/O PORT**
        ORAA   #8       **TO OUTPUT**
        STAA   MRFTA    **RX BURST DATA**
        STAA   LATCH3   RESET I/O #4 FIFO
        LDAA   SIGACQ
        ANDA   #07FH     SELECT I/O OUTPUT MODE
        STAA   SIGACQ
        STAA   LATCH1
IO4YC1  CLR    FMSDO4
        LDAA   #2
        STAA   MSDO4
        STAA   FLBFG4
        JMP    IO4G
IO4YD   LDAA   FLBFG4
        BEQ    IO4YF
        STAA   XFER4
        LDAA   CHCT4+1
        CMPA   #10      RCV ELASTIC BUFFER FULL?
        BGE    IO4YE
        JMP    BSERV
IO4YE   CLR    FLBFG4
        STAA   UT4
        LDAA   SIGACQ   SET OUTPUT MODE
        ANDA   #07FH
        STAA   SIGACQ
        STAA   LATCH1
        LDAA   MRFTA
        ANDA   #0F7H     RESET ON I/O FIFO
        STAA    MRFTA
        STAA    LATCH3
        JSR     LD4
        JSR     LD4       LOAD FIFO W/6 BUF. CH.
        LDAA    BPS16K
        BEQ     IO4YE0
        JSR     LD4       16 KBPS SO:
        JSR     LD4
        JSR     LD4
        JSR     LD4
        JSR     LD4       LOAD FIFO W/21 BUF. CH.
IO4YE0  JMP     BSERV
IO4YE1  LDAA    #1
        STAA    XFER4
        LDAA    UT4
        BNE     IO4YE0
        LDAA    DATA6     CHECK FIFO UNDERFLOW
        BITA    #1        BY TESTING FIFO "O.R."
        BEQ     IO4YC1    **UNDERFLOW**
        JMP     BSERV
IO4YF   LDX     CHCT4
        BEQ     IO4YE1
        LDAA    DATA6
        LDAB    UT4
        BNE     IO4YFA
        BITA    #1
        BEQ     IO4YC1
IO4YFA  BITA    #010H      CHECK "I.R.":  FULL?
        BEQ     BSERV
IO4YG   JSR     IO4LF     OUTPUT 1 CHAN. TO FIFO
        LDAA    DATA6
        BITA    #010H      CHECK "I.R.":  FULL?
        BEQ     BSERV
IO4YH   JSR     IO4LF     OUTPUT 2 CH. TO FIFO
        LDAA    DATA6
        BITA    #010H      CHECK "I.R.": FULL?
        BEQ     BSERV
IO4YI   JSR     IO4LF     OUTPUT 3 CH. TO FIFO
        LDAA    BPS16K
        BEQ     BSERV
IO4YJ   LDAA    DATA6     16KBPS:  SERVICE 10 CHAR.
        BITA    #010H      CHECK "I.R.": FULL?
        BEQ     BSERV
IO4YK   JSR     IO4LF     OUTPUT 4 CH. TO FIFO
        LDAA    DATA6
        BITA    #010H      CHECK "I.R.": FULL?
        BEQ     BSERV
IO4YL   JSR     IO4LF     OUTPUT 5 CH. TO FIFO
        LDAA    DATA6
        BITA    #010H      CHECK "I.R.": FULL?
        BEQ     BSERV
IO4YM   JSR     IO4LF     OUTPUT 6 CH. TO FIFO
        LDAA    DATA6
        BITA    #010H      CHECK "I.R.": FULL?
        BEQ     BSERV
IO4YN   JSR     IO4LF     OUTPUT 7 CH. TO FIFO
        LDAA    DATA6
        BITA   #010H      CHECK "I.R.": FULL?
        BEQ    BSERV
IO4YO   JSR    IO4LF     OUTPUT 8 CH. TO FIFO
        LDAA   DATA6
        BITA   #010H      CHECK "I.R.": FULL?
        BEQ    BSERV
IO4YP   JSR    IO4LF     OUTPUT 9 CH. TO FIFO
        LDAA   DATA6
        BITA   #010H      CHECK "I.R.": FULL?
IO4YQ   JSR    IO4LF     OUTPUT 10 CH. TO FIFO
**********************************************************
*
*   BITE SERVICE ROUTINE
*
*
BSERV   LDAA   DATA5
        BITA   #8        TEST:  SYSTEM BITE RESET
        BEQ    BSERVA
        CLR    IRSBIT
        CLR    BITELT
BSERVA  BITA   #04H       TEST:  LAMP TEST
        BEQ    BSERVB
        LDAA   #0FFH      TURN ALL BITE LITES ON
        BRA    BSERVC                                      WBR
*                         WHEN THE EXTEND TIMER IS SET,     WBR
*                         IT IS NOT AN ERROR FOR THE PORT   WBR
*                         OUTPUT TO BE DIFFERENT FROM THE   WBR
*                         FRAME INPUT, BECAUSE WE ARE       WBR
*                         DISABLING RECEIVE CAPABILITIES.   WBR
*                         THEREFORE WE MUST MASK OUT THE    WBR
*                         ERROR ONLY ON THE PORTS WITH      WBR
*                         THE EXTEND TIMER SET.  BITS 0-3   WBR
*                         IN RXINHB COORESPOND TO PORTS 1-4 WBR
*                         AND BITS 1-4 COORESPOND TO PORTS  WBR
*                         1-4 IN LATCH2.                    WBR
BSERVB  LDAA   RXINHB     NO ERROR FOR PORT WITH            WBR
        ASLA                 EXTEND TIMER SET               WBR
        COMA                                                WBR
        ANDA   BITELT     OUTPUT ALL OTHER ERRORS PRESENT   WBR
        STAA   BITELT                                       WBR
BSERVC  STAA   LATCH2                                       WBR
        LDAB   DATA5
        LDAA   TIMEFG
        BEQ    BSERVF
        BITB   #010H      TEST 75/16 HZ CLOCK
        BNE    BSERVE
BSERVD  CLR    TIMEFG
BSERVE  JMP    SOF
BSERVF  BITB   #010H      TEST 75/16 HZ CLOCK
        BEQ    BSERVE
        INC    TIMEFG
        LDAB   BITELT
        LDX    CHCT1
        BEQ    SERVF3
SERVF1  LDAA   XFER1
        BEQ    SERVF2
        CLR    XFER1
```

```
       BRA    BSERVG
SERVF2 ORAB   #2       I/O #1 BITE ERROR
       BRA    BSERVG
SERVF3 LDAA   XMTRQ1   NO DATA AFTER TX REQ?
       BNE    SERVF1
BSERVG LDX    CHCT2
       BEQ    SERVG3
SERVG1 LDAA   XFER2
       BEQ    SERVG2
       CLR    XFER2
       BRA    BSERVH
SERVG2 ORAB   #4       I/O #2 BITE ERROR
       BRA    BSERVH
SERVG3 LDAA   XMTRQ2   NO DATA AFTER TX REQ?
       BNE    SERVG1
BSERVH LDX    CHCT3
       BEQ    SERVH3
SERVH1 LDAA   XFER3
       BEQ    SERVH2
       CLR    XFER3
       BRA    BSERVI
SERVH2 ORAB   #8       I/O #3 BITE ERROR
       BRA    BSERVI
SERVH3 LDAA   XMTRQ3   NO DATA AFTER TX REQ?
       BNE    SERVH1
BSERVI LDX    CHCT4
       BEQ    SERVI3
SERVI1 LDAA   XFER4
       BEQ    SERVI2
       CLR    XFER4
       BRA    BSERVK
SERVI2 ORAB   #010H     I/O #4 BITE ERROR
       BRA    BSERVK
SERVI3 LDAA   XMTRQ4   NO DATA AFTER TX REQ?
       BNE    SERVI1
BSERVK STAB   BITELT
**********************************************************
*
*      BEGIN SERVICING SYSTEM OUTPUT FIFO
*
SOF    LDAA   SOEN
       BEQ    SOF1
       BSR    SOFA
SOF1   JMP    SIF
*
SOF2   LDX    OUTCHC   *TX SELF TEST ROUTINE*
       LDAB   #16
SOF3   LDAA   DATA5    FIFO RDY?
       BMI    SOF5
       LDAA   #033H     OUTPUT TEST CHAR.
       STAA   PIA1BD
       DEX
       BNE    SOF4
       CLRA            DONE SELF TEST TX
       STAA   SOEN
       STAA   SELFTT
       STX    OUTCHC
       RTS
SOF4   DECB
       BNE    SOF3
SOF5   STX    OUTCHC
       RTS
*
SOFA   LDAA   SELFTT    TX SELF TEST?
       BNE    SOF2
       LDX    TXPNTR
       LDAA   DATA5     CHECK TX INHIBIT
       BITA   #1
       BEQ    SOFAA
       CLR    SOEN      TX INHIBIT: ABORT
       RTS
SOFAA  LDX    2,X
       STX    TBPNTR
       LDX    TXPNTR
       LDX    4,X
       STX    TXCHCT
       LDX    TXPNTR
       LDX    14,X      SETUP L.G.D. PNTR FOR
       STX    TXLGDP    TX BURST SERVICE
       LDAA   GRP1FG
       BNE    SOFB
       JMP    SOFY
SOFB   LDX    FILLCT
       LDAB   #0FFH      FILL PATTERN BEFORE DATA
       LDAA   DATA5     FIFO RDY? (DATA5-H=LOW)
       BPL    SOFBA
       STX    FILLCT
       RTS
SOFBA  STAB   PIA1BD    OUTPUT FILL CHAR #1
       DEX
       BNE    SOFBB
       CLR    GRP1FG
       JMP    SOFD+2
SOFBB  LDAA   DATA5     FIFO READY?
       BPL    SOFBC
       STX    FILLCT
       RTS
SOFBC  STAB   PIA1BD    OUTPUT FILL CHAR #2
       DEX
       BNE    SOFBD
       CLR    GRP1FG
       JMP    SOFE+2
SOFBD  LDAA   DATA5     FIFO READY?
       BPL    SOFBE
       STX    FILLCT
       RTS
SOFBE  STAB   PIA1BD    OUTPUT FILL CHAR #3
       DEX
       BNE    SOFBF
       CLR    GRP1FG
       JMP    SOFF+2
SOFBF  LDAA   DATA5     FIFO READY?
       BPL    SOFBG
       STX    FILLCT
```

```
2957         RTS
2958 SOFBG   STAB    PIA1BD    OUTPUT FILL CHAR #4
2959         DEX
2960         BNE     SOFBH
2961         CLR     GRP1FG
2962         JMP     SOFG+2
2963 SOFBH   LDAA    DATA5     FIFO READY?
2964         BPL     SOFBI
2965         STX     FILLCT
2966         RTS
2967 SOFBI   STAB    PIA1BD    OUTPUT FILL CHAR #5
2968         DEX
2969         BNE     SOFBJ
2970         CLR     GRP1FG
2971         JMP     SOFH+2
2972 SOFBJ   LDAA    DATA5     FIFO READY?
2973         BPL     SOFBK
2974         STX     FILLCT
2975         RTS
2976 SOFBK   STAB    PIA1BD    OUTPUT FILL CHAR #6
2977         DEX
2978         BNE     SOFBL
2979         CLR     GRP1FG
2980         JMP     SOFI+2
2981 SOFBL   LDAA    DATA5     FIFO READY?
2982         BPL     SOFBM
2983         STX     FILLCT
2984         RTS
2985 SOFBM   STAB    PIA1BD    OUTPUT FILL CHAR #7
2986         DEX
2987         BNE     SOFBN
2988         CLR     GRP1FG
2989         JMP     SOFJ+2
2990 SOFBN   LDAA    DATA5     FIFO READY?
2991         BPL     SOFBO
2992         STX     FILLCT
2993         RTS
2994 SOFBO   STAB    PIA1BD    OUTPUT FILL CHAR #8
2995         DEX
2996         BNE     SOFBP
2997         CLR     GRP1FG
2998         JMP     SOFK+2
2999 SOFBP   LDAA    DATA5     FIFO READY?
3000         BPL     SOFBQ
3001         STX     FILLCT
3002         RTS
3003 SOFBQ   STAB    PIA1BD    OUTPUT FILL CHAR #9
3004         DEX
3005         BNE     SOFBR
3006         CLR     GRP1FG
3007         JMP     SOFL+2
3008 SOFBR   LDAA    DATA5     FIFO READY?
3009         BPL     SOFBS
3010         STX     FILLCT
3011         RTS
3012 SOFBS   STAB    PIA1BD    OUTPUT FILL CHAR #10
3013         DEX
```

```
3014         BNE    SOFBT
3015         CLR    GRP1FG
3016         JMP    SOFM+2
3017 SOFBT   LDAA   DATA5     FIFO READY?
3018         BPL    SOFBU
3019         STX    FILLCT
3020         RTS
3021 SOFBU   STAB   PIA1BD    OUTPUT FILL CHAR #11
3022         DEX
3023         BNE    SOFBV
3024         CLR    GRP1FG
3025         JMP    SOFN+2
3026 SOFBV   LDAA   DATA5     FIFO READY?
3027         BPL    SOFBW
3028         STX    FILLCT
3029         RTS
3030 SOFBW   STAB   PIA1BD    OUTPUT FILL CHAR #12
3031         DEX
3032         BNE    SOFBX
3033         CLR    GRP1FG
3034         JMP    SOFO+2
3035 SOFBX   LDAA   DATA5     FIFO READY?
3036         BPL    SOFBY
3037         STX    FILLCT
3038         RTS
3039 SOFBY   STAB   PIA1BD    OUTPUT FILL CHAR #13
3040         DEX
3041         BNE    SOFBZ
3042         CLR    GRP1FG
3043         JMP    SOFP+2
3044 SOFBZ   LDAA   DATA5     FIFO READY?
3045         BPL    SOFB1
3046         STX    FILLCT
3047         RTS
3048 SOFB1   STAB   PIA1BD    OUTPUT FILL CHAR #14
3049         DEX
3050         BNE    SOFB2
3051         CLR    GRP1FG
3052         JMP    SOFQ+2
3053 SOFB2   LDAA   DATA5     FIFO READY?
3054         BPL    SOFB3
3055         STX    FILLCT
3056         RTS
3057 SOFB3   STAB   PIA1BD    OUTPUT FILL CHAR #15
3058         DEX
3059         BNE    SOFB4
3060         CLR    GRP1FG
3061         JMP    SOFR+2
3062 SOFB4   LDAA   DATA5     FIFO READY?
3063         BPL    SOFB5
3064         STX    FILLCT
3065         RTS
3066 SOFB5   STAB   PIA1BD    OUTPUT FILL CHAR #16
3067         DEX
3068         BNE    SOFB6
3069         CLR    GRP1FG
3070 SOFB6   STX    FILLCT
```

```
        RTS
SOFC    LDAA   DATA5    FIFO READY?
        BPL    SOFCA
        JMP    SOFS
SOFCA   LDX    TXCHCT
        BNE    SOFCB
        LDAA   #0FFH     GET STUFF CHAR
        STAA   TXEND
        LDX    TXPNTR
        STAA   13,X     SET PORT'S OUFG
        BRA    SOFCD
SOFCB   DEX
        STX    TXCHCT
        LDX    TBPNTR
        LDAA   0,X      GET BUFF CHAR
        INX
        CPX    TXUVAL   ADJ BUFF PNTRS
        BNE    SOFCC
        LDX    TXLVAL
SOFCC   STX    TBPNTR
        CPX    TXLGDP   LAST GOOD DATA?
        BNE    SOFCD
        LDX    #0       YES
        STX    TXLGDP
        STAB   TXDONE
SOFCD   STAA   PIA1BD   OUTPUT CHAR #1
        LDX    OUTCHC
        DEX
        BNE    SOFD
        JMP    SOFT
SOFD    STX    OUTCHC
        LDAA   DATA5    FIFO READY?
        BPL    SOFDA
        JMP    SOFS
SOFDA   LDX    TXCHCT
        BNE    SOFDB
        LDAA   #0FFH     GET STUFF CHAR
        STAA   TXEND
        LDX    TXPNTR
        STAA   13,X
        BRA    SOFDD
SOFDB   DEX
        STX    TXCHCT
        LDX    TBPNTR
        LDAA   0,X      GET BUFF CHAR
        INX
        CPX    TXUVAL   ADJ BUFF PNTRS
        BNE    SOFDC
        LDX    TXLVAL
SOFDC   STX    TBPNTR
        CPX    TXLGDP   LAST GOOD DATA?
        BNE    SOFDD
        LDX    #0       YES
        STX    TXLGDP
        STAB   TXDONE
SOFDD   STAA   PIA1BD   OUTPUT CHAR #2
        LDX    OUTCHC
        DEX
        BNE    SOFE
        JMP    SOFT
SOFE    STX    OUTCHC
        LDAA   DATA5     FIFO READY?
        BPL    SOFEA
        JMP    SOFS
SOFEA   LDX    TXCHCT
        BNE    SOFEB
        LDAA   #0FFH      GET STUFF CHAR
        STAA   TXEND
        LDX    TXPNTR
        STAA   13,X      SET PORT'S UOFG
        BRA    SOFED
SOFEB   DEX
        STX    TXCHCT
        LDX    TBPNTR
        LDAA   0,X       GET BUFF CHAR
        INX
        CPX    TXUVAL    ADJ BUFF PNTRS
        BNE    SOFEC
        LDX    TXLVAL
SOFEC   STX    TBPNTR
        CPX    TXLGDP    LAST GOOD DATA?
        BNE    SOFED
        LDX    #0        YES
        STX    TXLGDP
        STAB   TXDONE
SOFED   STAA   PIA1BD    OUTPUT CHAR #3
        LDX    OUTCHC
        DEX
        BNE    SOFF
        JMP    SOFT
SOFF    STX    OUTCHC
        LDAA   DATA5     FIFO READY?
        BPL    SOFFA
        JMP    SOFS
SOFFA   LDX    TXCHCT
        BNE    SOFFB
        LDAA   #0FFH      GET STUFF CHAR
        STAA   TXEND
        LDX    TXPNTR
        STAA   13,X      SET PORT'S OUFG
        BRA    SOFFD
SOFFB   DEX
        STX    TXCHCT
        LDX    TBPNTR
        LDAA   0,X       GET BUFF CHAR
        INX
        CPX    TXUVAL    ADJ BUFF PNTRS
        BNE    SOFFC
        LDX    TXLVAL
SOFFC   STX    TBPNTR
        CPX    TXLGDP    LAST GOOD DATA?
        BNE    SOFFD
        LDX    #0        YES
        STX    TXLGDP
```

```
          STAB   TXDONE
SOFFD     STAA   PIA1BD    OUTPUT CHAR #4
          LDX    OUTCHC
          DEX
          BNE    SOFG
          JMP    SOFT
SOFG      STX    OUTCHC
          LDAA   DATA5     FIFO READY?
          BPL    SOFGA
          JMP    SOFS
SOFGA     LDX    TXCHCT
          BNE    SOFGB
          LDAA   #0FFH      GET STUFF CHAR
          STAA   TXEND
          LDX    TXPNTR
          STAA   13,X      SET PORT'S OUFG
          BRA    SOFGD
SOFGB     DEX
          STX    TXCHCT
          LDX    TBPNTR
          LDAA   0,X       GET BUFF CHAR
          INX
          CPX    TXUVAL    ADJ BUFF PNTRS
          BNE    SOFGC
          LDX    TXLVAL
SOFGC     STX    TBPNTR
          CPX    TXLGDP    LAST GOOD DATA?
          BNE    SOFGD
          LDX    #0        YES
          STX    TXLGDP
          STAB   TXDONE
SOFGD     STAA   PIA1BD    OUTPUT CHAR #5
          LDX    OUTCHC
          DEX
          BNE    SOFH
          JMP    SOFT
SOFH      STX    OUTCHC
          LDAA   DATA5     FIFO READY?
          BPL    SOFHA
          JMP    SOFS
SOFHA     LDX    TXCHCT
          BNE    SOFHB
          LDAA   #0FFH      GET STUFF CHAR
          STAA   TXEND
          LDX    TXPNTR
          STAA   13,X      SET PORT'S OUFG
          BRA    SOFHD
SOFHB     DEX
          STX    TXCHCT
          LDX    TBPNTR
          LDAA   0,X       GET BUFF CHAR
          INX
          CPX    TXUVAL    ADJ BUFF PNTRS
          BNE    SOFHC
          LDX    TXLVAL
SOFHC     STX    TBPNTR
          CPX    TXLGDP    LAST GOOD DATA?
```

```
         BNE   SOFHD
         LDX   #0       YES
         STX   TXLGDP
         STAB  TXDONE
SOFHD    STAA  PIA1BD   OUTPUT CHAR #6
         LDX   OUTCHC
         DEX
         BNE   SOFI
         JMP   SOFT
SOFI     STX   OUTCHC
         LDAA  DATA5    FIFO READY?
         BPL   SOFIA
         JMP   SOFS
SOFIA    LDX   TXCHCT
         BNE   SOFIB
         LDAA  #0FFH     GET STUFF CHAR
         STAA  TXEND
         LDX   TXPNTR
         STAA  13,X     SET PORT'S OUFG
         BRA   SOFID
SOFIB    DEX
         STX   TXCHCT
         LDX   TBPNTR
         LDAA  0,X      GET BUFF CHAR
         INX
         CPX   TXUVAL   ADJ BUFF PNTRS
         BNE   SOFIC
         LDX   TXLVAL
SOFIC    STX   TBPNTR
         CPX   TXLGDP   LAST GOOD DATA?
         BNE   SOFID
         LDX   #0       YES
         STX   TXLGDP
         STAB  TXDONE
SOFID    STAA  PIA1BD   OUTPUT CHAR #7
         LDX   OUTCHC
         DEX
         BNE   SOFJ
         JMP   SOFT
SOFJ     STX   OUTCHC
         LDAA  DATA5    FIFO READY?
         BPL   SOFJA
         JMP   SOFS
SOFJA    LDX   TXCHCT
         BNE   SOFJB
         LDAA  #0FFH     GET STUFF CHAR
         STAA  TXEND
         LDX   TXPNTR
         STAA  13,X     SET PORT'S OUFG
         BRA   SOFJD
SOFJB    DEX
         STX   TXCHCT
         LDX   TBPNTR
         LDAA  0,X      GET BUFF CHAR
         INX
         CPX   TXUVAL   ADJ BUFF PNTRS
         BNE   SOFJC
```

```
        LDX     TXLVAL
SOFJC   STX     TBPNTR
        CPX     TXLGDP    LAST GOOD DATA?
        BNE     SOFJD
        LDX     #0        YES
        STX     TXLGDP
        STAB    TXDONE
SOFJD   STAA    PIA1BD    OUTPUT CHAR #8
        LDX     OUTCHC
        DEX
        BNE     SOFK
        JMP     SOFT
SOFK    STX     OUTCHC
        LDAA    DATA5     FIFO READY?
        BPL     SOFKA
        JMP     SOFS
SOFKA   LDX     TXCHCT
        BNE     SOFKB
        LDAA    #0FFH      GET STUFF CHAR
        STAA    TXEND
        LDX     TXPNTR
        STAA    13,X      SET PORT'S OUFG
        BRA     SOFKD
SOFKB   DEX
        STX     TXCHCT
        LDX     TBPNTR
        LDAA    0,X       GET BUFF CHAR
        INX
        CPX     TXUVAL    ADJ BUFF PNTRS
        BNE     SOFKC
        LDX     TXLVAL
SOFKC   STX     TBPNTR
        CPX     TXLGDP    LAST GOOD DATA?
        BNE     SOFKD
        LDX     #0        YES
        STX     TXLGDP
        STAB    TXDONE
SOFKD   STAA    PIA1BD    OUTPUT CHAR #9
        LDX     OUTCHC
        DEX
        BNE     SOFL
        JMP     SOFT
SOFL    STX     OUTCHC
        LDAA    DATA5     FIFO READY?
        BPL     SOFLA
        JMP     SOFS
SOFLA   LDX     TXCHCT
        BNE     SOFLB
        LDAA    #0FFH      GET STUFF CHAR
        STAA    TXEND
        LDX     TXPNTR
        STAA    13,X      SET PORT'S OUFG
        BRA     SOFLD
SOFLB   DEX
        STX     TXCHCT
        LDX     TBPNTR
        LDAA    0,X       GET BUFF CHAR
```

```
         INX
         CPX    TXUVAL   ADJ BUFF PNTRS
         BNE    SOFLC
         LDX    TXLVAL
SOFLC    STX    TBPNTR
         CPX    TXLGDP   LAST GOOD DATA?
         BNE    SOFLD
         LDX    #0       YES
         STX    TXLGDP
         STAB   TXDONE
SOFLD    STAA   PIA1BD   OUTPUT CHAR #10
         LDX    OUTCHC
         DEX
         BNE    SOFM
         JMP    SOFT
SOFM     STX    OUTCHC
         LDAA   DATA5    FIFO READY?
         BPL    SOFMA
         JMP    SOFS
SOFMA    LDX    TXCHCT
         BNE    SOFMB
         LDAA   #0FFH     GET STUFF CHAR
         STAA   TXEND
         LDX    TXPNTR
         STAA   13,X     SET PORT'S OUFG
         BRA    SOFMD
SOFMB    DEX
         STX    TXCHCT
         LDX    TBPNTR
         LDAA   0,X      GET BUFF CHAR
         INX
         CPX    TXUVAL   ADJ BUFF PNTRS
         BNE    SOFMC
         LDX    TXLVAL
SOFMC    STX    TBPNTR
         CPX    TXLGDP   LAST GOOD DATA?
         BNE    SOFMD
         LDX    #0       YES
         STX    TXLGDP
         STAB   TXDONE
SOFMD    STAA   PIA1BD   OUTPUT CHAR #11
         LDX    OUTCHC
         DEX
         BNE    SOFN
         JMP    SOFT
SOFN     STX    OUTCHC
         LDAA   DATA5    FIFO READY?
         BPL    SOFNA
         JMP    SOFS
SOFNA    LDX    TXCHCT
         BNE    SOFNB
         LDAA   #0FFH     GET STUFF CHAR
         STAA   TXEND
         LDX    TXPNTR
         STAA   13,X     SET PORT' OUFG
         BRA    SOFND
SOFNB    DEX
```

```
         STX    TXCHCT
         LDX    TBPNTR
         LDAA   0,X      GET BUFF CHAR
         INX
         CPX    TXUVAL   AND BUFF PNTRS
         BNE    SOFNC
         LDX    TXLVAL
SOFNC    STX    TBPNTR
         CPX    TXLGDP   LAST GOOD DATA?
         BNE    SOFND
         LDX    #0       YES
         STX    TXLGDP
         STAB   TXDONE
SOFND    STAA   PIA1BD   OUTPUT CHAR #12
         LDX    OUTCHC
         DEX
         BNE    SOFO
         JMP    SOFT
SOFO     STX    OUTCHC
         LDAA   DATA5    FIFO READY?
         BPL    SOFOA
         JMP    SOFS
SOFOA    LDX    TXCHCT
         BNE    SOFOB
         LDAA   #0FFH     GET STUFF CHAR
         STAA   TXEND
         LDX    TXPNTR
         STAA   13,X     SET PORT'S OUFG
         BRA    SOFOD
SOFOB    DEX
         STX    TXCHCT
         LDX    TBPNTR
         LDAA   0,X      GET BUFF CHAR
         INX
         CPX    TXUVAL   ADJ BUFF PNTRS
         BNE    SOFOC
         LDX    TXLVAL
SOFOC    STX    TBPNTR
         CPX    TXLGDP   LAST GOOD DATA?
         BNE    SOFOD
         LDX    #0       YES
         STX    TXLGDP
         STAB   TXDONE
SOFOD    STAA   PIA1BD   OUTPUT CHAR #13
         LDX    OUTCHC
         DEX
         BNE    SOFP
         JMP    SOFT
SOFP     STX    OUTCHC
         LDAA   DATA5    FIFO READY?
         BPL    SOFPA
         JMP    SOFS
SOFPA    LDX    TXCHCT
         BNE    SOFPB
         LDAA   #0FFH     GET BUFF CHAR
         STAA   TXEND
         LDX    TXPNTR
```

```
        STAA   13,X      SET PORT'S OUFG
        BRA    SOFPD
SOFPB   DEX
        STX    TXCHCT
        LDX    TBPNTR
        LDAA   0,X       GET BUF CHAR
        INX
        CPX    TXUVAL    ADJ BUFF PNTRS
        BNE    SOFPC
        LDX    TXLVAL
SOFPC   STX    TBPNTR
        CPX    TXLGDP    LAST GOOD DATA?
        BNE    SOFPD
        LDX    #0        YES
        STX    TXLGDP
        STAB   TXDONE
SOFPD   STAA   PIA1BD    OUTPUT CHAR #14
        LDX    OUTCHC
        DEX
        BNE    SOFQ
        JMP    SOFT
SOFQ    STX    OUTCHC
        LDAA   DATA5     FIFO READY?
        BPL    SOFQA
        JMP    SOFS
SOFQA   LDX    TXCHCT
        BNE    SOFQB
        LDAA   #0FFH      GET STUFF CHAR
        STAA   TXEND
        LDX    TXPNTR
        STAA   13,X      SET PORT'S OUFG
        BRA    SOFQD
SOFQB   DEX
        STX    TXCHCT
        LDX    TBPNTR
        LDAA   0,X       GET BUFF CHAR
        INX
        CPX    TXUVAL    ADJ BUFF PNTRS
        BNE    SOFQC
        LDX    TXLVAL
SOFQC   STX    TBPNTR
        CPX    TXLGDP    LAST GOOD DATA?
        BNE    SOFQD
        LDX    #0        YES
        STX    TXLGDP
        STAB   TXDONE
SOFQD   STAA   PIA1BD    OUTPUT CHAR #15
        LDX    OUTCHC
        DEX
        BEQ    SOFT
SOFR    STX    OUTCHC
        LDAA   DATA5     FIFO READY?
        BPL    SOFRA
        JMP    SOFS
SOFRA   LDX    TXCHCT
        BNE    SOFRB
        LDAA   #0FFH      GET STUFF CHAR
```

```
       STAA   TXEND
       LDX    TXPNTR
       STAA   13,X      SET PORT'S OUFG
       BRA    SOFRD
SOFRB  DEX
       STX    TXCHCT
       LDX    TBPNTR
       LDAA   0,X       GET BUFF CHAR
       INX
       CPX    TXUVAL    ADJ BUFF PNTRS
       BNE    SOFRC
       LDX    TXLVAL
SOFRC  STX    TBPNTR
       CPX    TXLGDP    LAST GOOD DATA?
       BNE    SOFRD
       LDX    #0        YES
       STX    TXLGDP
       STAB   TXDONE
SOFRD  STAA   PIA1BD    OUTPUT CHAR #16
       LDX    OUTCHC
       DEX
       BEQ    SOFT
       STX    OUTCHC
SOFS   LDX    TXPNTR    STORE BUFF PNTRS
       LDAA   TXCHCT
       LDAB   TXCHCT+1
       STAA   4,X       RESTORE BUFF CHAR COUNT
       STAB   5,X
       LDAA   TBPNTR
       LDAB   TBPNTR+1
       STAA   2,X       RESTORE BUFF OUTPUT PNTR
       STAB   3,X
       LDAA   TXLGDP
       LDAB   TXLGDP+1
       STAA   14,X      RESTORE L.G.D. PNTR
       STAB   15,X
       RTS
SOFT   LDAA   TXEND
       BEQ    SOFTB
SOFTA  CLRB
       STAB   TXEND
       STAB   TXDONE
       STAB   TXCHCT
       STAB   TXCHCT+1
       LDAA   XMASK     CLR "DATA RDY PORT-"
       ANDA   PA1AD
       STAA   PA1AD
       STAA   PIA1AD
       LDX    TXPNTR
       STAB   8,X       CLR TX REQ FLAG
       STAB   13,X      CLR OUFG
       STAB   6,X       CLR EMPTY
       STAB   4,X       CLR CHCT
       STAB   5,X
       LDAB   #3
       STAB   7,X       SET FLBFG
       LDAA   MASKR     TX ACK OF I/O PORT
```

```
        ANDA   MRFTA    AND SET FIFO'S MR
        LDAB   XMASK
        COMB
        STAB   XMASK
        ORAA   XMASK
        STAA   MRFTA
        STAA   LATCH3
        BRA    SOFU
SOFTB   LDAA   TXDONE
        BEQ    SOFU
        LDAA   XMASK    TEST PORT'S TX REQ INPUT
        COMA
        ANDA   DATA1
        BEQ    SOFTA
SOFU    CLRA
        STAA   SOEN
        BRA    SOFS
SOFY    LDAB   #0FFH
        LDAA   TXEND
        BNE    SOFYA
        LDAA   TXDONE
        BEQ    SOFYXX
        LDAA   XMASK    TEST PORT'S TX REQ INPUT
        COMA
        ANDA   DATA1
        BEQ    SOFYXX   BRANCH IF TX REQ = 0
        CLRA            GET HERE WITH MULTIPLE
        STAA   TXDONE   TX REQS. PER FRAME
SOFYXX  JMP    SOFC
SOFYA   LDX    OUTCHC
        LDAA   DATA5    FIFO RDY?
        BPL    SOFYA1
        RTS
SOFYA1  STAB   PIA1BD   OUTPUT FILL CHAR #1
        DEX
        BEQ    SOFTA
SOFYB   LDAA   DATA5    FIFO RDY?
        BPL    SOFYB1
        STX    OUTCHC
        RTS
SOFYB1  STAB   PIA1BD   OUTPUT FILL CHAR #2
        DEX
        BEQ    SOFTA
SOFYC   LDAA   DATA5    FIFO RDY?
        BPL    SOFYC1
        STX    OUTCHC
        RTS
SOFYC1  STAB   PIA1BD   OUTPUT FILL CHAR #3
        DEX
        BNE    SOFYD
        JMP    SOFTA
SOFYD   LDAA   DATA5    FIFO RDY?
        BPL    SOFYD1
        STX    OUTCHC
        RTS
SOFYD1  STAB   PIA1BD   OUTPUT FILL CHAR #4
        DEX
        BNE    SOFYE
        JMP    SOFTA
SOFYE   LDAA   DATA5    FIFO RDY?
        BPL    SOFYE1
        STX    OUTCHC
        RTS
SOFYE1  STAB   PIA1BD   OUTPUT FILL CHAR #5
        DEX
        BNE    SOFYF
        JMP    SOFTA
SOFYF   LDAA   DATA5    FIFO RDY?
        BPL    SOFYF1
        STX    OUTCHC
        RTS
SOFYF1  STAB   PIA1BD   OUTPUT FILL CHAR #6
        DEX
        BNE    SOFYG
        JMP    SOFTA
SOFYG   LDAA   DATA5    FIFO RDY?
        BPL    SOFYG1
        STX    OUTCHC
        RTS
SOFYG1  STAB   PIA1BD   OUTPUT FILL CHAR #7
        DEX
        BNE    SOFYH
        JMP    SOFTA
SOFYH   LDAA   DATA5    FIFO RDY?
        BPL    SOFYH1
        STX    OUTCHC
        RTS
SOFYH1  STAB   PIA1BD   OUTPUT FILL CHAR #8
        DEX
        BNE    SOFYI
        JMP    SOFTA
SOFYI   LDAA   DATA5    FIFO RDY?
        BPL    SOFYI1
        STX    OUTCHC
        RTS
SOFYI1  STAB   PIA1BD   OUTPUT FILL CHAR #9
        DEX
        BNE    SOFYJ
        JMP    SOFTA
SOFYJ   LDAA   DATA5    FIFO RDY?
        BPL    SOFYJ1
        STX    OUTCHC
        RTS
SOFYJ1  STAB   PIA1BD   OUTPUT FILL CHAR #10
        DEX
        BNE    SOFYK
        JMP    SOFTA
SOFYK   LDAA   DATA5    FIFO RDY?
        BPL    SOFYK1
        STX    OUTCHC
        RTS
SOFYK1  STAB   PIA1BD   OUTPUT FILL CHAR #11
        DEX
        BNE    SOFYL
        JMP    SOFTA
SOFYL   LDAA   DATA5     FIFO RDY?
        BPL    SOFYL1
        STX    OUTCHC
        RTS
SOFYL1  STAB   PIA1BD    OUTPUT FILL CHAR #12
        DEX
        BNE    SOFYM
        JMP    SOFTA
SOFYM   LDAA   DATA5     FIFO RDY?
        BPL    SOFYM1
        STX    OUTCHC
        RTS
SOFYM1  STAB   PIA1BD    OUTPUT FILL CHAR #13
        DEX
        BNE    SOFYN
        JMP    SOFTA
SOFYN   LDAA   DATA5     FIFO RDY?
        BPL    SOFYN1
        STX    OUTCHC
        RTS
SOFYN1  STAB   PIA1BD    OUTPUT FILL CHAR #14
        DEX
        BNE    SOFYO
        JMP    SOFTA
SOFYO   LDAA   DATA5     FIFO RDY?
        BPL    SOFYO1
        STX    OUTCHC
        RTS
SOFYO1  STAB   PIA1BD    OUTPUT FILL CHAR #15
        DEX
        BNE    SOFYP
        JMP    SOFTA
SOFYP   LDAA   DATA5     FIFO RDY?
        BPL    SOFYP1
        STX    OUTCHC
        RTS
SOFYP1  STAB   PIA1BD    OUTPUT FILL CHAR #16
        DEX
        BNE    SOFYQ
        JMP    SOFTA
SOFYQ   STX    OUTCHC
        RTS
***********************************************************
*
*       BEGIN SERVICING SYSTEM INPUT FIFO
*
*
SIF     LDAA   SIEN
        BNE    SIFA
        JMP    TDMA
*
SIF1    LDX    INCHC     *RX SELF TEST ROUTINE*
        LDAB   #16
SIF2    LDAA   DATA1     CHAR IN RX FIFO?
        BPL    SIF6
        LDAA   DATA2     LOAD CHAR
```

```
       CMPA   #033H
       BEQ    SIF3     CORRECT CHARACTER?
       STAB   STERR
SIF3   DEX
       BNE    SIF5
       CLRA
       STAA   SIEN
       STAA   SELFTR
       STX    INCHC
       LDAA   PA1AD
       LDAB   STERR
       BNE    SIF3A
       ANDA   #07FH     CLR DBP SELF TEST FAIL
       BRA    SIF4
SIF3A  ORAA   #080H     SET DBP SELF TEST FAIL
SIF4   STAA   PA1AD
       STAA   PIA1AD
       ANDA   #080H     ADJUST SELF TEST
       ORAA   BITELT   FAIL BITE LITE
       STAA   BITELT
       STAA   LATCH2
       CLR    STERR
       JMP    TDMA
SIF5   DECB
       BNE    SIF2
SIF6   STX    INCHC
       JMP    TDMA
*
SIFA   LDAA   SELFTR   RX SELF TEST?
       BNE    SIF1
       LDAA   INPTRC
       BEQ    SIFA1A
       LDX    RCRXPT
SIFR   LDAA   DATA1    CHAR IN INPUT FIFO?
       BMI    SIFRA
SIFR1  STX    RCRXPT
       JMP    TDMA
SIFRA  LDAA   DATA2    LOAD & STORE FIFO CHAR
       STAA   0,X
       INX
       CPX    #ERCRX   BUFFER FULL?
       BNE    SIFR
       SEI
       LDX    #SRCRX
       LDAA   0,X
       STAA   ACIAD    LOAD ACIA WITH 1
       INX             R/CCOW RX CHARS.
       LDAA   #0BDH
       STAA   ACIACS
       STAA   TXIRFG
       STX    RCRXPT
       CLR    SIEN
       CLI
       JMP    TDMA
************************************************************
*
*FILL MISSING RX BURST WITH "FF" DATA
```

```
*
SIFRB  LDAB   #16       INIT STORE COUNTER
SIFRC  LDAA   #0FFH
       LDX    RBPNTR
       STAA   0,X
       INX              ADJ PNTRS
       CPX    RXUVAL
       BNE    SIFRD
       LDX    RXLVAL
SIFRD  STX    RBPNTR
       LDX    RBCHCT
       INX
       STX    RBCHCT
       LDX    INCHC     TEST FIFO CH CNT
       DEX
       BNE    SIFRE
       CLR    FILLFG
       JMP    SIFQ
SIFRE  STX    INCHC
       DECB             DEC STORE COUNTER
       BNE    SIFRC
       JMP    SIFP
*
*
SIFA1A LDX    RXPNTR    SETUP RCV PNTRS
       LDAA   8,X       TEST PORT'S TX REQ FLAG
       BNE    SIFA1C
       LDAA   PTONRX
       ANDA   MASKIO    TEST IF PORT IS ON
       BNE    SIFA1B
SIFA1C CLR    SIEN      *ABORT RX BURST:  TX REQ*
       CLR    FILLFG    *OR PORT IS OFF*
       JMP    TDMA
SIFA1B LDX    0,X
       STX    RBPNTR
       LDX    RXPNTR
       LDX    4,X
       STX    RBCHCT
       LDAA   FILLFG    FILL BURST WITH "FF"?
       BNE    SIFRB
       LDAA   DATA1
       BMI    SIFA1
       JMP    TDMA
SIFA1  LDAA   DATA2     LOAD & STORE CHAR #1
       LDX    RBPNTR
       STAA   0,X
       INX              ADJ PNTRS
       CPX    RXUVAL
       BNE    SIFA2
       LDX    RXLVAL
SIFA2  STX    RBPNTR
       LDX    RBCHCT
       INX
       STX    RBCHCT
       LDX    INCHC     TEST FIFO CH CNT
       DEX
       BNE    SIFB
```

```
        JMP    SIFQ
SIFB    STX    INCHC
        LDAA   DATA1
        BMI    SIFB1
        JMP    SIFP
SIFB1   LDAA   DATA2     LOAD & STORE FIFO CHAR #2
        LDX    RBPNTR
        STAA   0,X
        INX              ADJ PNTRS
        CPX    RXUVAL
        BNE    SIFB2
        LDX    RXLVAL
SIFB2   STX    RBPNTR
        LDX    RBCHCT
        INX
        STX    RBCHCT
        LDX    INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE    SIFC
        JMP    SIFQ
SIFC    STX    INCHC
        LDAA   DATA1
        BMI    SIFC1
        JMP    SIFP
SIFC1   LDAA   DATA2     LOAD & STORE FIFO CHAR #3
        LDX    RBPNTR
        STAA   0,X
        INX              ADJ PNTRS
        CPX    RXUVAL
        BNE    SIFC2
        LDX    RXLVAL
SIFC2   STX    RBPNTR
        LDX    RBCHCT
        INX
        STX    RBCHCT
        LDX    INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE    SIFD
        JMP    SIFQ
SIFD    STX    INCHC
        LDAA   DATA1
        BMI    SIFD1
        JMP    SIFP
SIFD1   LDAA   DATA2     LOAD & STORE FIFO CHAR #4
        LDX    RBPNTR
        STAA   0,X
        INX              ADJ PNTRS
        CPX    RXUVAL
        BNE    SIFD2
        LDX    RXLVAL
SIFD2   STX    RBPNTR
        LDX    RBCHCT
        INX
        STX    RBCHCT
        LDX    INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE    SIFE
        JMP    SIFQ
SIFE    STX    INCHC
        LDAA   DATA1
        BMI    SIFE1
        JMP    SIFP
SIFE1   LDAA   DATA2     LOAD & STORE FIFO CHAR #5
        LDX    RBPNTR
        STAA   0,X
        INX              ADJ PNTRS
        CPX    RXUVAL
        BNE    SIFE2
        LDX    RXLVAL
SIFE2   STX    RBPNTR
        LDX    RBCHCT
        INX
        STX    RBCHCT
        LDX    INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE    SIFF
        JMP    SIFQ
SIFF    STX    INCHC
        LDAA   DATA1
        BMI    SIFF1
        JMP    SIFP
SIFF1   LDAA   DATA2     LOAD & STORE FIFO CHAR #6
        LDX    RBPNTR
        STAA   0,X
        INX              ADJ PNTRS
        CPX    RXUVAL
        BNE    SIFF2
        LDX    RXLVAL
SIFF2   STX    RBPNTR
        LDX    RBCHCT
        INX
        STX    RBCHCT
        LDX    INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE    SIFG
        JMP    SIFQ
SIFG    STX    INCHC
        LDAA   DATA1
        BMI    SIFG1
        JMP    SIFP
SIFG1   LDAA   DATA2     LOAD & STORE FIFO CHAR #7
        LDX    RBPNTR
        STAA   0,X
        INX              ADJ PNTRS
        CPX    RXUVAL
        BNE    SIFG2
        LDX    RXLVAL
SIFG2   STX    RBPNTR
        LDX    RBCHCT
        INX
        STX    RBCHCT
        LDX    INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE    SIFH
        JMP    SIFQ
SIFH    STX    INCHC
        LDAA   DATA1
        BMI    SIFH1
        JMP    SIFP
SIFH1   LDAA   DATA2     LOAD & STORE FIFO CHAR #8
        LDX    RBPNTR
        STAA   0,X
        INX              ADJ PNTRS
        CPX    RXUVAL
        BNE    SIFH2
        LDX    RXLVAL
SIFH2   STX    RBPNTR
        LDX    RBCHCT
        INX
        STX    RBCHCT
        LDX    INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE    SIFI
        JMP    SIFQ
SIFI    STX    INCHC
        LDAA   DATA1
        BMI    SIFI1
        JMP    SIFP
SIFI1   LDAA   DATA2     LOAD & STORE FIFO CHAR #9
        LDX    RBPNTR
        STAA   0,X
        INX              ADJ PNTRS
        CPX    RXUVAL
        BNE    SIFI2
        LDX    RXLVAL
SIFI2   STX    RBPNTR
        LDX    RBCHCT
        INX
        STX    RBCHCT
        LDX    INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE    SIFX
        JMP    SIFQ
SIFX    STX    INCHC
        LDAA   DATA1
        BMI    SIFX1
        JMP    SIFP
SIFX1   LDAA   DATA2     LOAD & STORE FIFO CHAR #10
        LDX    RBPNTR
        STAA   0,X
        INX              ADJ PNTRS
        CPX    RXUVAL
        BNE    SIFX2
        LDX    RXLVAL
SIFX2   STX    RBPNTR
        LDX    RBCHCT
        INX
        STX    RBCHCT
        LDX    INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE    SIFJ
        JMP   SIFQ
SIFJ    STX   INCHC
        LDAA  DATA1
        BMI   SIFJ1
        JMP   SIFP
SIFJ1   LDAA  DATA2     LOAD & STORE FIFO CHAR #11
        LDX   RBPNTR
        STAA  0,X
        INX
        CPX   RXUVAL    ADJ PNTRS
        BNE   SIFJ2
        LDX   RXLVAL
SIFJ2   STX   RBPNTR
        LDX   RBCHCT
        INX
        STX   RBCHCT
        LDX   INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE   SIFK
        JMP   SIFQ
SIFK    STX   INCHC
        LDAA  DATA1
        BMI   SIFK1
        JMP   SIFP
SIFK1   LDAA  DATA2     LOAD & STORE FIFO CHAR #12
        LDX   RBPNTR
        STAA  0,X
        INX             ADJ PNTRS
        CPX   RXUVAL
        BNE   SIFK2
        LDX   RXLVAL
SIFK2   STX   RBPNTR
        LDX   RBCHCT
        INX
        STX   RBCHCT
        LDX   INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE   SIFL
        JMP   SIFQ
SIFL    STX   INCHC
        LDAA  DATA1
        BMI   SIFL1
        JMP   SIFP
SIFL1   LDAA  DATA2     LOAD & STORE FIFO CHAR #13
        LDX   RBPNTR
        STAA  0,X
        INX             ADJ PNTRS
        CPX   RXUVAL
        BNE   SIFL2
        LDX   RXLVAL
SIFL2   STX   RBPNTR
        LDX   RBCHCT
        INX
        STX   RBCHCT
        LDX   INCHC     TEST FIFO CHAR COUNT
        DEX
        BNE   SIFM
```

```
       JMP    SIFQ
SIFM   STX    INCHC
       LDAA   DATA1
       BPL    SIFP
SIFM1  LDAA   DATA2    LOAD & STORE FIFO CHAR #14
       LDX    RBPNTR
       STAA   0,X
       INX             ADJ PNTRS
       CPX    RXUVAL
       BNE    SIFM2
       LDX    RXLVAL
SIFM2  STX    RBPNTR
       LDX    RBCHCT
       INX
       STX    RBCHCT
       LDX    INCHC    TEST FIFO CHAR COUNT
       DEX
       BEQ    SIFQ
SIFN   STX    INCHC
       LDAA   DATA1
       BPL    SIFP
SIFN1  LDAA   DATA2    LOAD & STORE FIFO CHAR #15
       LDX    RBPNTR
       STAA   0,X
       INX             ADJ PNTRS
       CPX    RXUVAL
       BNE    SIFN2
       LDX    RXLVAL
SIFN2  STX    RBPNTR
       LDX    RBCHCT
       INX
       STX    RBCHCT
       LDX    INCHC    TEST FIFO CHAR COUNT
       DEX
       BEQ    SIFQ
SIFO   STX    INCHC
       LDAA   DATA1
       BPL    SIFP
SIFO1  LDAA   DATA2    LOAD & STORE FIFO CHAR #16
       LDX    RBPNTR
       STAA   0,X
       INX             ADJ PNTRS
       CPX    RXUVAL
       BNE    SIFO2
       LDX    RXLVAL
SIFO2  STX    RBPNTR
       LDX    RBCHCT
       INX
       STX    RBCHCT
       LDX    INCHC    TEST FIFO CHAR COUNT
       DEX
       BEQ    SIFQ
       STX    INCHC
SIFP   LDX    RXPNTR   UPDATE PORT'S BUFF PNTRS.
       LDAA   RBCHCT
       LDAB   RBCHCT+1
       STAA   4,X
```

```
        STAB   5,X
        LDAA   RBPNTR
        LDAB   RBPNTR+1
        STAA   0,X
        STAB   1,X
        JMP    TDMA
SIFQ    CLRA
        STAA   SIEN
        BRA    SIFP
************************************************************
*
*   INTERRUPT REQUEST SERVICE ROUTINE
*
*
IRS     LDAB   ACIACS
        LDAA   TXIRFG
        BEQ    IRSC
        BITB   #2        TDR EMPTY?
        BEQ    IRSC
        LDX    RCRXPT
IRSA    LDAA   0,X
        STAA   ACIAD     OUTPUT CHAR TO ACIA
        INX
        STX    RCRXPT
        CPX    #ERCRX
        BNE    IRSB
        LDAA   #09DH      CLR TX IRQ ENABLE
        STAA   ACIACS
        CLR    TXIRFG
IRSB    BITB   #1        RDR FULL?
        BNE    IRSE
        RTI
IRSC    BITB   #1
        BNE    IRSE
IRSD    LDAA   #1        RCCOW/CCOW BITE ERROR:
        STAA   IRSBIT    IRQ ERROR OR SCU FORMAT ERR
        RTI
IRSE    BITB   #070H      ACIA RX ERRORS?
        BEQ    IRSF
        LDAA   #1
        STAA   IRSBIT
IRSF    LDAA   ACIAD     LOAD ACIA RX CHAR
        LDAB   RCFG
        BEQ    IRSJ
        DECB
        BEQ    IRSG
        DECB
        BNE    IRSN
        LDX    RTXPT
        STAA   0,X       STORE CHAR IN RCCOW BUFFER
        INX
        STX    RTXPT
        CPX    #ERTXB
        BNE    IRSI
        LDAA   #1        SET FOR RCCOW DATA READY
        STAA   RDRDY
        BRA    IRSH
```

```
IRSG    LDX     CTXPT
        STAA    0,X       STORE CHAR IN CCOW BUFFER
        INX
        STX     CTXPT
        CPX     #ECTXB
        BNE     IRSI
        LDAA    #2        SET FOR CCOW DATA RDY
        STAA    CDRDY
IRSH    CLR     RCFG
IRSI    RTI
IRSJ    CMPA    #0A3H
        BNE     IRSL
        LDX     #SCTXB
        STX     CTXPT
        LDAB    #1
IRSK    STAB    RCFG
        RTI
IRSL    CMPA    #0A4H
        BNE     IRSM
        LDX     #SRTXB
        STX     RTXPT
        LDAB    #2
        BRA     IRSK
IRSM    CMPA    #0A9H
        BEQ     IRSMA
        CMPA    #0ABH                                            WBR
        BEQ     IRSMA                                            WBR
        BRA     IRSD                                             WBR
IRSMA   TAB              GET RXINHB BIT 2 FROM A9/A11 BIT 2      WBR
        ANDB    #02H                                             WBR
        STAB    RXINHB                                           WBR
        LDAB    #3
        STAB    RC3FG
        BRA     IRSK
IRSN    LDAB    RC3FG
        CMPB    #3
        BEQ     IRSQ
        CMPB    #2
        BEQ     IRSO
        TAB               GET RXINHB BITS 4,3 FROM BITS 7,6      WBR
        RORB                                                     WBR
        RORB                                                     WBR
        RORB                                                     WBR
        RORB                                                     WBR
        ANDB    #0CH                                             WBR
        ORAB    RXINHB                                           WBR
        STAB    RXINHB                                           SEF
        ANDA    #03FH     REMOVE RXINHB BITS BEFORE USING        WBR
        STAA    BRATE2    PORT BIT RATES (3 & 4)
        TAB
        ANDB    #7
        ANDA    #038H
        ASLB
        ASLB
        ASLB
        STAB    CHRAT3
        STAA    CHRAT4
```

```
         CLR     RCFG
         CLR     RC3FG
         RTI
IRSO     TAB               GET RXINHB BIT 0 FROM BIT 6          WBR
         ROLB                                                   WBR
         ROLB                                                   WBR
         ROLB                                                   WBR
         ANDB    #01H                                           WBR
         ORAB    RXINHB                                         WBR
         STAB    RXINHB                                         SEF
         ANDA    #0BFH     REMOVE RXINHB BIT BEFORE USING       WBR
         STAA    BRATE1    PORT BIT RATES (1 & 2)
         TAB
         ANDB    #080H                                          WBR
         STAB    BPS16K
         TAB
         ANDB    #7
         ANDA    #038H
         ASLB
         ASLB
         ASLB
         STAB    CHRAT1
         STAA    CHRAT2
IRSP     DEC     RC3FG
         RTI
IRSQ     STAA    PTONRX    PORT ON & RX ONLY INFO
         BRA     IRSP
*************************************************************
*
* LOAD RX ELASTIC BUFFER FOR 16 KBPS
*
STUFF    LDX     #SBUFF1
         CLRB
STUFA    STAA    0,X
         INX
         INCB
         BNE     STUFA
         RTS
*************************************************************
*
INSERR BRA       INSERR    BAD INTERRUPT- CRASH
*
*************************************************************
*
* PROGRAM VERSION, ROM CHECKSUM VALUES, AND INTERRUPT VECTORS.
*
*************************************************************
         ORG     0FFF5H
PGMVRS FCB       4         **PROGRAM VERSION**
CKSMB    RMB     1         MSB OF CHECKSUM - BD. PROGRAMMER SUPPLIES VALUE
CKSMA    RMB     1         LSB OF CHECKSUM - SEE 98-P08293T FOR DETAILS.
         FDB     IRS       IRQ VECTOR
         FDB     INSERR    SWI ERROR
         FDB     INSERR    NMI ERROR
         FDB     PWRON     RESTART VECTOR
*
         END     PWRON
```

APPENDIX E

A. DATA BUFFER PROCESSOR LABEL CROSS REFERENCE

Label Name	Defined On	Referenced On
BITRAT	642	637 943

BSERV	2801	2351 2385 2406 2524 2528 2566 2580 2662
		2684 2720 2740 2748 2757 2761 2765 2768
		2771 2775 2779 2783 2787 2791

BSERVA	2806	2803
BSERVB	2821	2807
BSERVC	2826	2809
BSERVD	2832	
BSERVE	2833	2831 2835
BSERVF	2834	2829
BSERVG	2848	2843 2845
BSERVH	2858	2853 2855
BSERVI	2868	2863 2865
BSERVK	2878	2873 2875
CKSMA	4318	
CKSMB	4317	
CLEAR	116	29 126
CLEARA	118	120
INIT	68	11 175
INSERR	4308	4308
IO1A	1128	988
IO1A1	989	986
IO1A2	992	990
IO1A3	1025	1023
IO1AA	1132	1130
IO1B	1136	1134
IO1BA	1144	1147
IO1BB	1145	1143
IO1C	1148	1137 1139
IO1CA	1152	1141
IO1D	1160	1149 1314
IO1E	1164	1151
IO1F	1168	1166
IO1G	1200	1342
IO1H	1210	1161
IO1HA	1214	1212
IO1HB	1217	1215
IO1I	1221	1219 1284
IO1K	1226	1224

Label Name	Defined On	Referenced On							
IO1LF	1065	1387	1391	1395	1401	1405	1409	1413	1417
		1421	1425						
IO1LF1	1071	1068							
IO1LFA	1078	1073							
IO1LFB	1081	1075	1077	1079					
IO1LFC	1095	1085	1089						
IO1LFD	1102	1098							
IO1LS	1030	1221	1226	1231	1236	1244	1249	1254	1259
		1264	1269	1274					
IO1LSA	1053	1048							
IO1LSB	1056	1050	1052	1054					
IO1LSX	1047	1037							
IO1M	1231	1229							
IO1N10	1274	1272							
IO1N2	1236	1234							
IO1N3	1240	1238							
IO1N4	1244	1242							
IO1N5	1249	1247							
IO1N6	1254	1252							
IO1N7	1259	1257							
IO1N8	1264	1262							
IO1N9	1269	1267							
IO1O	1276	1216							
IO1P	1279								
IO1PT	984	712	729	739	778	816	854	927	932
		976							
IO1XP	1285	1220	1225	1230	1235	1239	1243	1248	1253
		1258	1263	1268	1273	1275	1278		
IO1XQ	1291	1286							
IO1Y	1292	1131	1135						
IO1YA	1314	1303	1316						
IO1YA1	1318	1213							
IO1YB	1315	1293							
IO1YC	1330								
IO1YC1	1338	1376	1384						
IO1YD	1343	1144	1317						
IO1YE	1350	1348							
IO1YE0	1369	1363	1373						
IO1YE1	1370	1101	1379						
IO1YF	1378	1344							
IO1YFA	1385	1382							
IO1YG	1387								
IO1YH	1391								
IO1YI	1395								
IO1YJ	1398								

Label Name	Defined On	Referenced On
IO1YK	1401	
IO1YL	1405	
IO1YM	1409	
IO1YN	1413	
IO1YO	1417	
IO1YP	1421	
IO1YQ	1425	
IO2A	1574	1434
IO2A1	1435	1432
IO2A2	1438	1436
IO2A3	1471	1469
IO2AA	1578	1576
IO2B	1582	1580
IO2BA	1590	1593
IO2BB	1591	1589
IO2C	1594	1583 1585
IO2CA	1598	1587
IO2D	1606	1595 1771
IO2DA	1609	1607
IO2E	1611	1597
IO2F	1615	1613
IO2FA	1639	1634
IO2FB	1642	1638
IO2G	1653	1799
IO2GA	1657	1654
IO2GB	1658	1656
IO2H	1667	1608
IO2HA	1671	1669
IO2HB	1674	1672
IO2I	1678	1676 1741
IO2K	1683	1681
IO2LF	1511	1844 1848 1852 1858 1862 1866 1870 1874 1878 1882
IO2LF1	1517	1514
IO2LFA	1524	1519
IO2LFB	1527	1521 1523 1525
IO2LFC	1541	1531 1535
IO2LFD	1548	1544
IO2LS	1476	1678 1683 1688 1693 1701 1706 1711 1716 1721 1726 1731
IO2LSA	1499	1494
IO2LSB	1502	1496 1498 1500
IO2LSX	1493	1483
IO2M	1688	1686
IO2N10	1731	1729
IO2N2	1693	1691

Label Name	Defined On	Referenced On							
IO2N3	1697	1695							
IO2N4	1701	1699							
IO2N5	1706	1704							
IO2N6	1711	1709							
IO2N7	1716	1714							
IO2N8	1721	1719							
IO2N9	1726	1724							
IO2O	1733	1673							
IO2P	1736								
IO2PT	1430	991	1025	1046	1163	1167	1199	1209	1291
		1313	1349	1369	1377	1386	1390	1394	1397
		1400	1404	1408	1412	1416	1420	1424	
IO2XP	1742	1677	1682	1687	1692	1696	1700	1705	1710
		1715	1720	1725	1730	1732	1735		
IO2XQ	1748	1743							
IO2Y	1749	1577	1581						
IO2YA	1771	1760	1773						
IO2YA1	1775	1670							
IO2YB	1772	1750							
IO2YC	1787								
IO2YC1	1795	1833	1841						
IO2YD	1800	1590	1774						
IO2YE	1807	1805							
IO2YE0	1826	1820	1830						
IO2YE1	1827	1547	1836						
IO2YF	1835	1801							
IO2YFA	1842	1839							
IO2YG	1844								
IO2YH	1848								
IO2YI	1852								
IO2YJ	1855								
IO2YK	1858								
IO2YL	1862								
IO2YM	1866								
IO2YN	1870								
IO2YO	1874								
IO2YP	1878								
IO2YQ	1882								
IO3A	2031	1891							
IO3A1	1892	1889							
IO3A2	1895	1893							
IO3A3	1928	1926							
IO3AA	2035	2033							
IO3B	2039	2037							
IO3BA	2047	2050							
IO3BB	2048	2046							

Label Name	Defined On	Referenced On							
IO3C	2051	2040	2042						
IO3CA	2055	2044							
IO3D	2063	2052	2228						
IO3DA	2066	2064							
IO3E	2068	2054							
IO3F	2072	2070							
IO3FA	2096	2091							
IO3FB	2099	2095							
IO3G	2110	2256							
IO3GA	2114	2111							
IO3GB	2115	2113							
IO3H	2124	2065							
IO3HA	2128	2126							
IO3HB	2131	2129							
IO3I	2135	2133	2198						
IO3K	2140	2138							
IO3LF	1968	2301	2305	2309	2315	2319	2323	2327	2331
		2335	2339						
IO3LF1	1974	1971							
IO3LFA	1981	1976							
IO3LFB	1984	1978	1980	1982					
IO3LFC	1998	1988	1992						
IO3LFD	2005	2001							
IO3LS	1933	2135	2140	2145	2150	2158	2163	2168	2173
		2178	2183	2188					
IO3LSA	1956	1951							
IO3LSB	1959	1953	1955	1957					
IO3LSX	1950	1940							
IO3M	2145	2143							
IO3N10	2188	2186							
IO3N2	2150	2148							
IO3N3	2154	2152							
IO3N4	2158	2156							
IO3N5	2163	2161							
IO3N6	2168	2166							
IO3N7	2173	2171							
IO3N8	2178	2176							
IO3N9	2183	2181							
IO3O	2190	2130							
IO3P	2193								
IO3PT	1887	1437	1471	1492	1610	1614	1652	1666	1748
		1770	1806	1826	1834	1843	1847	1851	1854
		1857	1861	1865	1869	1873	1877	1881	
IO3XP	2199	2134	2139	2144	2149	2153	2157	2162	2167
		2172	2177	2182	2187	2189	2192		
IO3XQ	2205	2200							

Label Name	Defined On	Referenced On
IO3Y	2206	2034 2038
IO3YA	2228	2217 2230
IO3YA1	2232	2127
IO3YB	2229	2207
IO3YC	2244	
IO3YC1	2252	2290 2298
IO3YD	2257	2047 2231
IO3YE	2264	2262
IO3YE0	2283	2277 2287
IO3YE1	2284	2004 2293
IO3YF	2292	2258
IO3YFA	2299	2296
IO3YG	2301	
IO3YH	2305	
IO3YI	2309	
IO3YJ	2312	
IO3YK	2315	
IO3YL	2319	
IO3YM	2323	
IO3YN	2327	
IO3YO	2331	
IO3YP	2335	
IO3YQ	2339	
IO4A	2488	2348
IO4A1	2349	2346
IO4A2	2352	2350
IO4A3	2385	2383
IO4AA	2492	2490
IO4B	2496	2494
IO4BA	2504	2507
IO4BB	2505	2503
IO4C	2508	2497 2499
IO4CA	2512	2501
IO4D	2520	2509 2685
IO4DA	2523	2521
IO4E	2525	2511
IO4F	2529	2527
IO4FA	2553	2548
IO4FB	2556	2552
IO4G	2567	2713
IO4GA	2571	2568
IO4GB	2572	2570
IO4H	2581	2522
IO4HA	2585	2583
IO4HB	2588	2586
IO4I	2592	2590 2655

Label Name	Defined On	Referenced On							
IO4K	2597	2595							
IO4LF	2425	2758	2762	2766	2772	2776	2780	2784	2788
		2792	2795						
IO4LF1	2431	2428							
IO4LFA	2438	2433							
IO4LFB	2441	2435	2437	2439					
IO4LFC	2455	2445	2449						
IO4LFD	2462	2458							
IO4LS	2390	2592	2597	2602	2607	2615	2620	2625	2630
		2635	2640	2645					
IO4LSA	2413	2408							
IO4LSB	2416	2410	2412	2414					
IO4LSX	2407	2397							
IO4M	2602	2600							
IO4N10	2645	2643							
IO4N2	2607	2605							
IO4N3	2611	2609							
IO4N4	2615	2613							
IO4N5	2620	2618							
IO4N6	2625	2623							
IO4N7	2630	2628							
IO4N8	2635	2633							
IO4N9	2640	2638							
IO4O	2647	2587							
IO4P	2650								
IO4PT	2344	1894	1928	1949	2067	2071	2109	2123	2205
		2227	2263	2283	2291	2300	2304	2308	2311
		2314	2318	2322	2326	2330	2334	2338	
IO4XP	2656	2591	2596	2601	2606	2610	2614	2619	2624
		2629	2634	2639	2644	2646	2649		
IO4XQ	2662	2657							
IO4Y	2663	2491	2495						
IO4YA	2685	2674	2687						
IO4YA1	2689	2584							
IO4YB	2686	2664							
IO4YC	2701								
IO4YC1	2709	2747	2755						
IO4YD	2714	2504	2688						
IO4YE	2721	2719							
IO4YE0	2740	2734	2744						
IO4YE1	2741	2461	2750						
IO4YF	2749	2715							
IO4YFA	2756	2753							
IO4YG	2758								
IO4YH	2762								
IO4YI	2766								

Label Name	Defined On	Referenced On						
IO4YJ	2769							
IO4YK	2772							
IO4YL	2776							
IO4YM	2780							
IO4YN	2784							
IO4YO	2788							
IO4YP	2792							
IO4YQ	2795							
IRS	4168							
IRSA	4174							
IRSB	4183	4179						
IRSC	4186	4170	4172					
IRSD	4188	4238						
IRSE	4191	4184	4187					
IRSF	4195	4192						
IRSG	4211	4199						
IRSH	4219	4210						
IRSI	4220	4207	4216					
IRSJ	4221	4197						
IRSK	4226	4233	4244					
IRSL	4228	4222						
IRSM	4234	4229						
IRSMA	4239	4235	4237					
IRSN	4245	4201						
IRSO	4271	4249						
IRSP	4291	4294						
IRSQ	4293	4247						
LD1	1107	1360	1361	1364	1365	1366	1367	1368
LD2	1553	1817	1818	1821	1822	1823	1824	1825
LD3	2010	2274	2275	2278	2279	2280	2281	2282
LD4	2467	2731	2732	2735	2736	2737	2738	2739
PGMVRS	4316							
PWRON	6							
PWRON1	58	30						
RAMERR	31	19	34					
RAMT	16	27						
RAMTST	17	22						
RBI	195	186	900					
RBI1C	225	222						
RBI1CA	228	226						
RBI1D	232	230						
RBI1DA	241	237						
RBI1E	242	234						
RBI1EA	252	247						
RBI1G	264	245						

Label Name	Defined On	Referenced On	
RBI1GA	271		
RBI1GB	284	278	
RBI1H	293	266	283
RBI1H1	294	263	
RBI1HA	303	297	
RBI1HB	307	302	
RBI2B	313	223	
RBI2C	316	314	
RBI2CA	319	317	
RBI2D	323	321	
RBI2DA	332	328	
RBI2E	333	325	
RBI2EA	343	338	
RBI2G	355	336	
RBI2GA	362		
RBI2GB	377	369	
RBI2H	386	357	376
RBI2H1	387	354	
RBI2HA	396	390	
RBI2HB	400	395	
RBI3B	406	315	
RBI3C	409	407	
RBI3CA	412	410	
RBI3D	416	414	
RBI3DA	425	421	
RBI3E	426	418	
RBI3EA	436	431	
RBI3G	448	429	
RBI3GA	455		
RBI3GB	470	462	
RBI3H	479	450	469
RBI3H1	480	447	
RBI3HA	489	483	
RBI3HB	493	488	
RBI4B	499	408	
RBI4C	502	500	
RBI4CA	505	503	
RBI4D	509	507	
RBI4DA	518	514	
RBI4E	519	511	
RBI4EA	529	524	
RBI4G	541	522	
RBI4GA	548		
RBI4GB	563	555	
RBI4H	572	543	562

Label Name	Defined On	Referenced On			
RBI4H1	573	540			
RBI4HA	582	576			
RBI4HB	586	581			
RBIA	198	196			
RBIB	210	199			
RBIBA	217	213			
RBIJ	594	501			
RBIJA	600	595			
RBIK	610	602			
RBIKA	616	634			
RBIL	619	614			
RBIM	625	636			
RBIN	631	612			
RBIO	635	632			
RBIS	637	311	404	497	590
RBISA	651	647			
RBISB	656	652			
RBISC	661	657			
RBISD	666	662			
RBISE	671	667			
RBISF	674	643			
RBIT	683	599	638		
RBIU	677	645			
ROMERR	53	48	50	56	
ROMTSA	41	46			
ROMTST	38	28			
SERVF1	2840	2847			
SERVF2	2844	2841			
SERVF3	2846	2839			
SERVG1	2850	2857			
SERVG2	2854	2851			
SERVG3	2856	2849			
SERVH1	2860	2867			
SERVH2	2864	2861			
SERVH3	2866	2859			
SERVI1	2870	2877			
SERVI2	2874	2871			
SERVI3	2876	2869			
SETUP	126				
SETUP1	127	58	188		
SETUP3	152	141			
SIF	3746	2886			
SIF1	3750	3784			
SIF2	3752	3779			
SIF3	3758	3756			

Label Name	Defined On	Referenced On
SIF3A	3769	3766
SIF4	3770	3768
SIF5	3778	3759
SIF6	3780	3753
SIFA	3783	3747
SIFA1	3855	3853
SIFA1A	3836	3786
SIFA1B	3845	3841
SIFA1C	3842	3838
SIFA2	3862	3860
SIFB	3870	3868
SIFB1	3874	3872
SIFB2	3881	3879
SIFC	3889	3887
SIFC1	3893	3891
SIFC2	3900	3898
SIFD	3908	3906
SIFD1	3912	3910
SIFD2	3919	3917
SIFE	3927	3925
SIFE1	3931	3929
SIFE2	3938	3936
SIFF	3946	3944
SIFF1	3950	3948
SIFF2	3957	3955
SIFG	3965	3963
SIFG1	3969	3967
SIFG2	3976	3974
SIFH	3984	3982
SIFH1	3988	3986
SIFH2	3995	3993
SIFI	4003	4001
SIFI1	4007	4005
SIFI2	4014	4012
SIFJ	4041	4039
SIFJ1	4045	4043
SIFJ2	4052	4050
SIFK	4060	4058
SIFK1	4064	4062
SIFK2	4071	4069
SIFL	4079	4077
SIFL1	4083	4081
SIFL2	4090	4088
SIFM	4098	4096
SIFM1	4101	

Label Name	Defined On	Referenced On							
SIFM2	4108	4106							
SIFN	4115								
SIFN1	4118								
SIFN2	4125	4123							
SIFO	4132								
SIFO1	4135								
SIFO2	4142	4140							
SIFP	4150	3833	3873	3892	3911	3930	3949	3968	3987
		4006	4025	4044	4063	4082	4100	4117	4134
		4162							
SIFQ	4160	3829	3869	3888	3907	3926	3945	3964	3983
		4002	4021	4040	4059	4078	4097	4114	4131
		4148							
SIFR	3788	3796							
SIFR1	3790								
SIFRA	3792	3789							
SIFRB	3813	3851							
SIFRC	3814	3832							
SIFRD	3821	3819							
SIFRE	3830	3827							
SIFX	4022	4020							
SIFX1	4026	4024							
SIFX2	4033	4031							
SOF	2883	2833							
SOF1	2886	2884							
SOF2	2888	2907							
SOF3	2890	2902							
SOF4	2901	2895							
SOF5	2903	2891							
SOFA	2906	2885							
SOFAA	2914	979	2911						
SOFB	2925	2923							
SOFB1	3048	3045							
SOFB2	3053	3050							
SOFB3	3057	3054							
SOFB4	3062	3059							
SOFB5	3066	3063							
SOFB6	3070	3068							
SOFBA	2931	2928							
SOFBB	2936	2933							
SOFBC	2940	2937							
SOFBD	2945	2942							
SOFBE	2949	2946							
SOFBF	2954	2951							
SOFBG	2958	2955							

Label Name	Defined On	Referenced On	
SOFBH	2963	2960	
SOFBI	2967	2964	
SOFBJ	2972	2969	
SOFBK	2976	2973	
SOFBL	2981	2978	
SOFBM	2985	2982	
SOFBN	2990	2987	
SOFBO	2994	2991	
SOFBP	2999	2996	
SOFBQ	3003	3000	
SOFBR	3008	3005	
SOFBS	3012	3009	
SOFBT	3017	3014	
SOFBU	3021	3018	
SOFBV	3026	3023	
SOFBW	3030	3027	
SOFBX	3035	3032	
SOFBY	3039	3036	
SOFBZ	3044	3041	
SOFC	3072	3612	
SOFCA	3075	3073	
SOFCB	3082	3076	
SOFCC	3090	3088	
SOFCD	3096	3081	3092
SOFD	3101	3099	
SOFDA	3105	3103	
SOFDB	3112	3106	
SOFDC	3120	3118	
SOFDD	3126	3111	3122
SOFE	3131	3129	
SOFEA	3135	3133	
SOFEB	3142	3136	
SOFEC	3150	3148	
SOFED	3156	3141	3152
SOFF	3161	3159	
SOFFA	3165	3163	
SOFFB	3172	3166	
SOFFC	3180	3178	
SOFFD	3186	3171	3182
SOFG	3191	3189	
SOFGA	3195	3193	
SOFGB	3202	3196	
SOFGC	3210	3208	
SOFGD	3216	3201	3212
SOFH	3221	3219	

Label Name	Defined On	Referenced On	
SOFHA	3225	3223	
SOFHB	3232	3226	
SOFHC	3240	3238	
SOFHD	3246	3231	3242
SOFI	3251	3249	
SOFIA	3255	3253	
SOFIB	3262	3256	
SOFIC	3270	3268	
SOFID	3276	3261	3272
SOFJ	3281	3279	
SOFJA	3285	3283	
SOFJB	3292	3286	
SOFJC	3300	3298	
SOFJD	3306	3291	3302
SOFK	3311	3309	
SOFKA	3315	3313	
SOFKB	3322	3316	
SOFKC	3330	3328	
SOFKD	3336	3321	3332
SOFL	3341	3339	
SOFLA	3345	3343	
SOFLB	3352	3346	
SOFLC	3360	3358	
SOFLD	3366	3351	3362
SOFM	3371	3369	
SOFMA	3375	3373	
SOFMB	3382	3376	
SOFMC	3390	3388	
SOFMD	3396	3381	3392
SOFN	3401	3399	
SOFNA	3405	3403	
SOFNB	3412	3406	
SOFNC	3420	3418	
SOFND	3426	3411	3422
SOFO	3431	3429	
SOFOA	3435	3433	
SOFOB	3442	3436	
SOFOC	3450	3448	
SOFOD	3456	3441	3452
SOFP	3461	3459	
SOFPA	3465	3463	
SOFPB	3472	3466	
SOFPC	3480	3478	
SOFPD	3486	3471	3482
SOFQ	3491	3489	

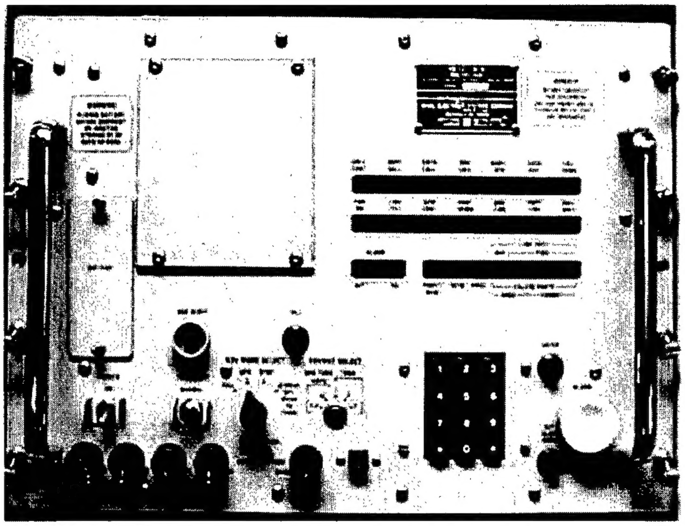
Label Name	Defined On	Referenced On							
SOFQA	3495	3493							
SOFQB	3502	3496							
SOFQC	3510	3508							
SOFQD	3516	3501	3512						
SOFR	3520								
SOFRA	3524	3522							
SOFRB	3531	3525							
SOFRC	3539	3537							
SOFRD	3545	3530	3541						
SOFS	3550	3074	3104	3134	3164	3194	3224	3254	3284
		3314	3344	3374	3404	3434	3464	3494	3523
		3600							
SOFT	3564	3100	3130	3160	3190	3220	3250	3280	3310
		3340	3370	3400	3430	3460	3490	3519	3548
SOFTA	3566	3597	3619	3626	3634	3642	3650	3658	3666
		3674	3682	3690	3698	3706	3714	3722	3730
		3738							
SOFTB	3592	3565							
SOFU	3598	3591	3593						
SOFY	3601	2924							
SOFYA	3613	3603							
SOFYA1	3617	3615							
SOFYB	3620								
SOFYB1	3624	3621							
SOFYC	3627								
SOFYC1	3631	3628							
SOFYD	3635	3633							
SOFYD1	3639	3636							
SOFYE	3643	3641							
SOFYE1	3647	3644							
SOFYF	3651	3649							
SOFYF1	3655	3652							
SOFYG	3659	3657							
SOFYG1	3663	3660							
SOFYH	3667	3665							
SOFYH1	3671	3668							
SOFYI	3675	3673							
SOFYI1	3679	3676							
SOFYJ	3683	3681							
SOFYJ1	3687	3684							
SOFYK	3691	3689							
SOFYK1	3695	3692							
SOFYL	3699	3697							
SOFYL1	3703	3700							
SOFYM	3707	3705							

Label Name	Defined On	Referenced On							
SOFYM1	3711	3708							
SOFYN	3715	3713							
SOFYN1	3719	3716							
SOFYO	3723	3721							
SOFYO1	3727	3724							
SOFYP	3731	3729							
SOFYP1	3735	3732							
SOFYQ	3739	3737							
SOFYXX	3612	3605	3609						
STUFA	4301	4304							
STUFF	4299	279	370	463	556				
TBI	692	197	227	231	251	318	322	342	411
		415	435	504	508	528	617	630	
TBI1	698	693							
TBI1BA	741	738							
TBI1E	748	742							
TBI1EA	752	747							
TBI1F	768	766							
TBI2	703	697							
TBI2B	773	735							
TBI2BA	779	777							
TBI2E	786	780							
TBI2EA	790	785							
TBI2F	806	804							
TBI3	705	699							
TBI3B	811	774							
TBI3BA	817	815							
TBI3E	824	818							
TBI3EA	828	823							
TBI3F	844	842							
TBI4	710	706							
TBI4B	849	812							
TBI4BA	855	853							
TBI4E	862	856							
TBI4EA	866	861							
TBI4F	882	880							
TBIA	713	711							
TBIB	721	714							
TBIBA	730	728							
TBIG	887	730	850						
TBIG1	897	899							
TBIGA	901	888							
TBIGB	908	910							
TBIGC	912	905							
TBIH	914	918							

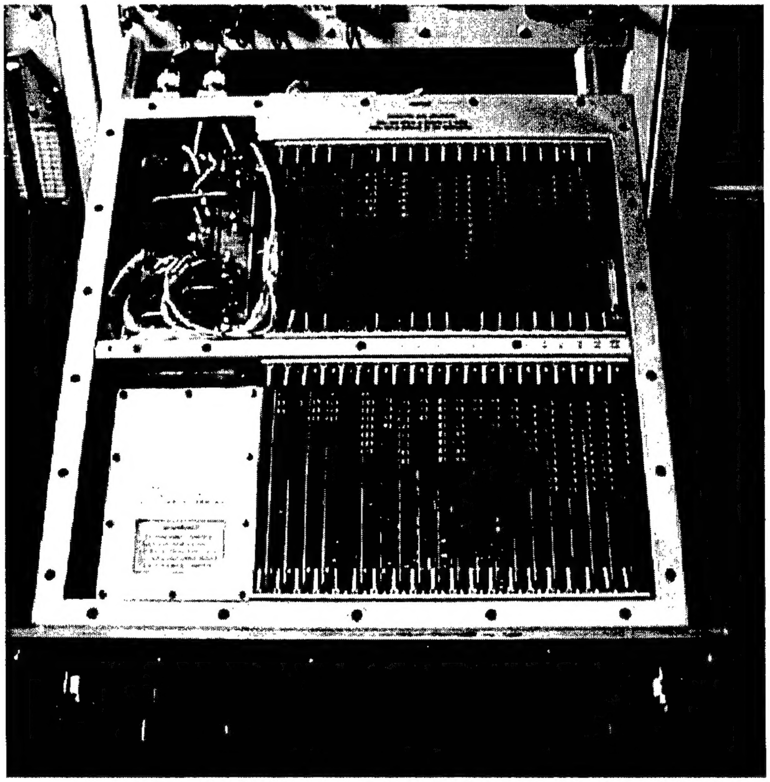
Label Name	Defined On	Referenced On
TBIHA	921	942
TBIHB	925	911
TBIHC	928	926
TBIJ	933	902
TBIK	935	939
TBIN	943	771 809 847 885
TBIO	944	
TBIQ	949	
TBIQ1	952	
TBIR	962	958
TBIRA	964	959 961
TBIRB	977	970
TBIS	979	951 963
TDMA	156	64 172 183 3748 3777 3781 3791 3808 3844 3854 4159
TDMA1	185	159
TDMAA	170	167
TDMAB	171	169
TDMAI	174	161
TIBA	731	725
TSTROM	28	24

APPENDIX F

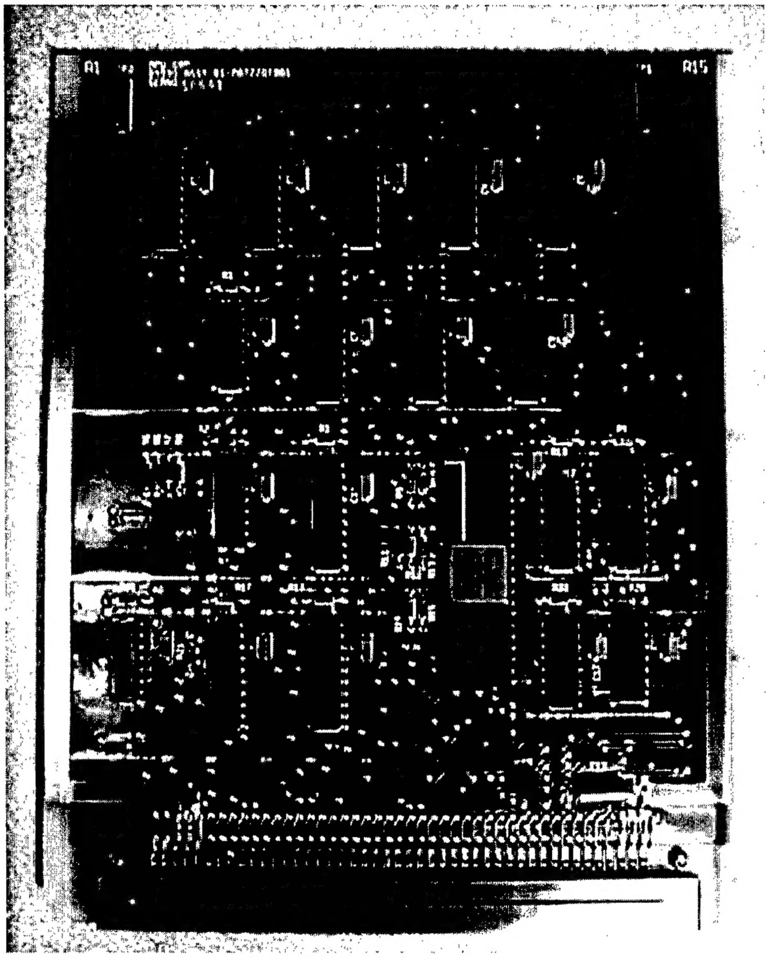
A. PHOTOS



TD-1271B/U Front Panel



TD-1271B/U Card Cage



DBP Processor Card

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman RD., Ste 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5150

3. Center for Naval Analysis 1
   4401 Ford Ave.
   Alexandria, VA 22302-0268

4. Chief of Naval Research 1
   800 North Quincy St.
   Arlington, VA 22217

5. Dr. Dan Boger, Chairman, Code CS 1
   Computer Science Dept.
   Naval Postgraduate School
   Monterey, CA 93943-5100

6. Dr. Man-Tak Shing, Code CS/SH 1
   Computer Science Dept.
   Naval Postgraduate School
   Monterey, CA 93943-5100

7. Kathleen M. Nelson 1
   SPAWAR SYSCEN SAN DIEGO
   D844
   San Diego, CA 92152-5150

8. Laurence Nixon 1
   SPAWAR SYSCEN SAN DIEGO
   D832
   San Diego, CA 92152-5150

9. Joe Hirschfelder 1
   SPAWAR SYSCEN SANDIEGO
   D874
   San Diego, CA 92152-5150